UNIVERSIDAD PÚBLICA DE NAVARRA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

PROYECTO FINAL DE CARRERA

Titulación:

INGENIERO DE TELECOMUNICACIÓN

Título del proyecto:

Representación de variables eléctricas en el sistema embebido Mini2440

Autor:

Fernando Marcotegui Zabalza

Tutor:

Dr. Jesús Corres Sanz

Pamplona, 2011

Universidad Pública de Navarra

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

PROYECTO FINAL DE CARRERA

Titulación: Ingeniero de Telecomunicación

Título: Representación de variables eléctricas en el sistema embebido Mini2440

Autor: Fernando Marcotegui Zabalza

Tutor: Dr. Jesús Corres Sanz

Pamplona, 2011

Ella está en el horizonte. Me acerco dos pasos, ella se aleja dos pasos.
Camino diez pasos y el horizonte se desplaza diez pasos más allá.
Por mucho que camine, nunca la alcanzaré. Entonces, ¿para qué sirve la utopía?
Para eso: sirve para caminar.

Eduardo Galeano

Declaración de autoría

Declaro que soy autor de este trabajo de Proyecto Final de Carrera y autorizo a la Universidad Pública de Navarra y a la Universidad de Pinar del Río a hacer uso del mismo con la finalidad que estimen conveniente.

_____

Fernando Marcotegui Zabalza

////////////////////////////////////

Agradecimientos

A Román, por su inestimable y desinteresada colaboración; sin ella este trabajo no hubiera sido posible.

A mis tutores Wilfredo y Jesús, a Rolando, a Bárbaro y a todas las personas que han dedicado a este proyecto parte de su precioso tiempo.

A mis padres y a mi familia, por su amor y apoyo incondicionales.

A Javi, Alberto, Yoany y familia, Wilfredo y familia, Eric y familia, a Raciel, Robe, Carlitos, Ledi, Jardiel, Yudemir, Luisito, Roger y su mamá, Any Ester, a sus maridos, esposas y amistades, a la familia Valdivia, a Carlos de Holguín y su familia, a Liudmila, Maite y Raúl de La Habana, al Cosi y a Wilfredo de Viñales, al grupo Unison, a los frikis de la rockoteca de Pinar del Río, a la gente de la sede, y a todas las personas que me han acompañado en las peripecias vividas durante estos meses inolvidables y me han brindado una amistad y una hospitalidad inmensas.

Al movimiento del software libre y del conocimiento libre en general, por mantener vivos mis sueños y los de muchas otras personas.

A Lisset, por enseñarme que el conocimiento es atemorizante, pero mucho más terrible es vivir sin él.

Mi más profundo agradecimiento a todos.

Índice

INTRODUCCIÓN

El presente trabajo final de carrera surge de un proyecto de colaboración entre la Universidad de Pinar del Río (UPR) en Cuba y la Universidad Pública de Navarra (UPNA) en España, como consecuencia del interés de ambas universidades en la investigación en el área de los nanosensores. Una de las líneas de trabajo que está siguiendo la UPNA pretende utilizar los conocimientos adquiridos en el desarrollo de nanosensores basados en fibra óptica para su aplicación en la detección de parámetros biomédicos, tales como la humedad relativa en la respiración o el nivel de glucosa en la sangre.

Esta línea de investigación se puede incluir, pues, en el campo de la ingeniería biomédica y particularmente en el del desarrollo de dispositivos médicos, el cual está cobrando una creciente importancia en los últimos años: se estima que en 2006 el mercado global de los dispositivos médicos superó los doscientos mil millones de dólares estadounidenses, presentando una esperanza de crecimiento anual de entre el 6 y el 9% hasta 2010. Por otra parte, es de destacar el gran interés que tienen para Cuba los sistemas y aplicaciones propios, desarrollados dentro del país y con "denominación de origen" nacional, considerando el modelo económico del país y las particulares características de su desarrollo industrial.

En el contexto de esta colaboración surgió la idea de desarrollar un sistema con una interfaz de usuario para la monitorización en tiempo real de variables biomédicas, que permita ser utilizado como herramienta de apoyo para el diagnóstico de ciertas patologías (por ejemplo, enfermedades respiratorias utilizando sensores de humedad relativa en la respiración). Se convino dividir dicho proyecto en tres partes:

i) En primer lugar se debe realizar el diseño del sistema de medida basado en nanosensores. Esta parte del proyecto comprende el diseño opto-electrónico de acondicionamiento de la señal y la fabricación de las placas de circuito impreso necesarias.

ii) En segundo lugar, se desarrollará la herramienta software de monitorización para la interfaz con el usuario, partiendo de la señal proveniente del sensor, una vez acondicionada a tal efecto.

iii) Por último, se desarrollarán los sistemas acabados para la utilización en las aplicaciones mencionadas de detección de parámetros biomédicos.

Así, el presente trabajo de tesis se incluye en la segunda de las tres partes del proyecto arriba descritas. No obstante, dado que la implementación de la herramienta de monitorización es independiente del resto del proyecto, es posible abstraerse del origen biomédico de la señal que debe ser monitorizada y extender el objeto de este trabajo a cualquier señal eléctrica, siempre y cuando dicha señal posea las características de las señales acondicionadas de acuerdo con la primera parte del proyecto. Esto es interesante tanto para la UPR como para la UPNA, ya que permite ampliar los resultados obtenidos en este trabajo a otros campos de aplicación, no exclusivamente dentro del proyecto original. En definitiva, el **problema** es la *necesidad de representación de variables eléctricas* en un sistema de detección de parámetros biomédicos.

Una vez definido el problema, el siguiente paso fue determinar la tecnología sobre la cual se implementaría la herramienta de monitorización. Se consideraron diversas posibilidades, como la utilización de equipos médicos ya existentes en el mercado, osciloscopios, sistemas embebidos o la configuración *ad hoc* de una FPGA, por nombrar algunas. Finalmente se concluyó que se emplearía un sistema embebido, al presentar interesantes ventajas sobre el resto de alternativas. En primer lugar, su coste es muy inferior a un equipo médico o a un osciloscopio digital. Por otra parte, los sistemas embebidos tienen la característica de integrar funcionalidades de una computadora personal en un espacio reducido y en un entorno compatible con otras aplicaciones. Esto proporciona ciertas ventajas, como la posibilidad de ir añadiendo funcionalidades a la herramienta o incluso nuevas herramientas de forma modular. En el caso de los sistemas biomédicos, por poner un ejemplo, se podría vincular las aplicaciones del dispositivo con una base de datos que contenga el historial clínico del paciente. Además, evidentemente, el tamaño reducido de estos dispositivos facilita el manejo y la portabilidad para este tipo de aplicaciones.

El sistema embebido que se decidió adquirir es la placa de desarrollo Mini2440 de FriendlyARM, que cuenta con un microprocesador Samsung S3C2440 de arquitectura ARM, una pantalla gráfica LCD táctil y un conversor analógico-digital de 4 canales y 10 bits de resolución, entre otros subsistemas.

Por lo tanto, la **hipótesis** que se plantea en este trabajo de tesis es que *es posible resolver el problema* descrito anteriormente *utilizando el dispositivo embebido Mini2440*. El **objetivo** es, pues, *diseñar e implementar una aplicación para dicho dispositivo que represente gráficamente señales eléctricas en el dominio del tiempo*. Para ello se consideró necesario cumplir los siguientes **objetivos específicos**:

- Análisis del estado del arte y estudio de las diferentes alternativas y técnicas
- Definición del entorno de desarrollo, eligiendo las herramientas software a utilizar
- Construcción del entorno de desarrollo, configurando los dispositivos involucrados y estableciendo la comunicación entre ellos
- Desarrollo y depuración de la herramienta para la arquitectura x86
- Desarrollo y depuración de la herramienta para la arquitectura ARM
- Descarga de la aplicación final a la arquitectura ARM e integración en su plataforma

Para llevar a cabo el diseño y la implementación de este proyecto se emplearon diferentes métodos en busca de una coherencia organizativa. Se utilizaron los métodos empíricos de observación científica y experimentación. Éstos posibilitaron verificar, progresivamente, el proceso de desarrollo de la herramienta. Los métodos teóricos de análisis histórico-lógico, modelación, hipotético-deductivo y de abstracción facilitaron la interpretación conceptual de los datos recogidos.

La computación, las técnicas de acceso a la información y a la comunicación y la revisión bibliográfica correspondiente permitieron conocer el estado del arte en las materias tratadas. La programación de código en lenguaje C++ haciendo uso de las herramientas de desarrollo de la biblioteca Qt y el manejo de las herramientas de compilación, gestión de recursos y comunicación del entorno GNU/Linux, fueron las principales técnicas empleadas para cumplir con los objetivos trazados.

1. SISTEMAS EMBEBIDOS

1.1 ¿Qué es un sistema embebido[1]?

Dar una descripción precisa de qué es un sistema embebido puede resultar una tarea complicada. El experto en software embebido Michael Barr, en su publicación *Programming Embedded Systems in C and C++* (O'Reilly, 1999), definía, no sin cierta ambigüedad, un sistema embebido como:

> *una combinación de hardware y software de computadora, sumado tal vez a algunas piezas mecánicas o de otro tipo, diseñada para tener una función específica.*

Esta definición se centra en la idea de que los sistemas embebidos en principio son diseñados específicamente para llevar a cabo de manera eficiente una tarea en particular, de tal forma que los recursos utilizados son optimizados para resolver dicho problema en concreto. Por otra parte, de esta definición se desprende que en el sistema debe haber tanto hardware como software de computadora, por lo tanto debe existir un sistema de computación gobernado por un microprocesador o similares[2].

Así, un sistema embebido se encontraría en un punto intermedio entre una computadora de propósito general –como por ejemplo una computadora personal (PC), que está diseñada para permitir realizar, idealmente, la mayor variedad posible de tareas– y un sistema que es diseñado por hardware específicamente para una tarea o un conjunto reducido de ellas, como por ejemplo los sistemas tradicionales de control, los dispositivos de lógica programable (PLDs) o las *field-programmable gate arrays* (FPGAs), que son programados directamente en hardware.

No obstante, en los últimos años han cobrado importancia algunos tipos de dispositivos que, si bien no encajan estrictamente en la definición anterior, comparten ciertas características con el resto de los sistemas embebidos y por tanto resulta razonable

1 Estos sistemas son conocidos como "embebidos" o "empotrados" . Ambos términos hacen referencia al hecho que la electrónica o el sistema electrónico de control es una parte integral del sistema en que se encuentra. Aquí se utilizará siempre el término "embebido".

2 Como se explicará en la sección 1.3.2, en lugar de un microprocesador podría haber un microcontrolador o un DSP.

incluirlos en este grupo. En concreto, los *smartphones* y las computadoras *tablets*, así como las *personal digital assistants* (PDAs), son dispositivos que podríamos situar a caballo entre las PCs y los sistemas embebidos, pues aunque no están diseñados para realizar una única función específica, presentan muchas características típicas de estos últimos, como las restricciones en los recursos computacionales disponibles o la manera de desarrollar el software .

Ante esta cuestión, el citado Michael Barr, desde su perspectiva de desarrollador de software, en la actualidad es partidario de incluir estos últimos dispositivos mencionados en el conjunto de los sistemas embebidos. Siguiendo el mismo razonamiento, en este trabajo se considerará que la placa de desarrollo Mini2440 de FriendlyARM, con la cual se llevó a cabo este proyecto, es también un sistema embebido a todos los efectos, ya que su arquitectura es muy similar a una *tablet* o a una PDA (véase el capítulo 2 para una descripción detallada de la placa Mini2440).

1.2 Algunos ejemplos

Es posible encontrar infinidad de ejemplos de dispositivos que contienen sistemas embebidos y que son empleados en la actualidad a diario en el llamado “primer mundo”, en la esfera doméstica, profesional, industrial, científica, militar... Además de los ya mencionados *smartphones*, *tablets* y PDAs, a continuación se citan algunos otros:

- Relojes digitales: en cualquier dispositivo donde haya un reloj digital existe un sistema embebido, tanto en un reloj de muñeca como en un horno o en un termómetro digital, por ejemplo. Típicamente, un reloj digital contiene un procesador sencillo de 4 u 8 bits (empleado principalmente para permitir dar soporte con el mismo hardware a una variedad de modelos y características) y su propia memoria ROM integrada.

- Electrodomésticos: en prácticamente cualquier electrodoméstico fabricado en las últimas décadas se encuentran sistemas embebidos, desde una batidora a una lavadora o a una tostadora. Cada tipo de electrodoméstico dispondrá de un sistema específico que le permita realizar su función e interacturar con el usuario si es necesario.

- Automóviles: el control electrónico de los automóviles fabricados hoy en día es cada vez más complejo y precisa de numerosos dispositivos embebidos, para los sistemas de ignición, transmisión, frenado, suspensión, control de tracción, dirección asistida, seguridad, localización geográfica o control de emisiones. Un automóvil puede tener hasta un centenar de microprocesadores y microcontroladores, muchos de ellos con comunicación entre sí.

- Máquinas de producción industrial: en los procesos industriales, el control de motores, hornos, maquinaria, etc. es manejado por sistemas embebidos, los cuales a menudo ofrecen un interfaz persona–máquina para ser dirigidos por un operario e informarle al mismo de la marcha del proceso.

- Videoconsolas: en muchos casos, este tipo de máquinas tienen más potencia de procesamiento que los PCs de su generación y su procesador está altamente especializado para las demandas del tipo de videojuegos que va a permitir jugar. En ocasiones el diseño de cada procesador corre a cargo del propio fabricante de la videoconsola.

- Sistemas radar de aviones: los dispositivos para el procesado de la señal recibida o reflejada de los sistemas radar embarcados en aviones requieren alta potencia de cálculo además de ocupar poco espacio, pesar poco y soportar condiciones extremas de funcionamiento (temperatura, presión atmosférica, vibraciones, etc.).

Así, se podría enumerar un sinfín más de ejemplos: teléfonos móviles, reproductores de audio o video, periféricos de una PC, navegadores GPS, controles remotos, equipos médicos, cajeros automáticos, máquinas expendedoras...

1.3 Características de los sistemas embebidos

1.3.1 Características generales

Como se desprende de lo expuesto en el apartado 1.1, los sistemas embebidos resultan más flexibles y más versátiles que los dispositivos diseñados totalmente por hardware, pero disponen de recursos más limitados que los sistemas de propósito general.

Esta limitación puede afectar a diversas características del producto final y depende de las especificaciones del problema y de la solución concretos. Por un lado tiene impacto sobre el hardware del sistema, y como consecuencia también sobre su software. Estas características se tratarán en las próximas dos secciones.

Por otro lado, además, aunque esto no es exclusivo de los sistemas embebidos, estos sistemas a menudo están sujetos en mayor o menor medida a cumplir requisitos de:

- Tamaño y/o peso: en el caso de un reloj digital de muñeca, por ejemplo, el sistema debe ser lo suficientemente pequeño y ligero como para ser transportado con comodidad. Aquí se consigue cumplir con dichos requisitos integrando un procesador sencillo y la memoria ROM en el mismo chip, y no utilizando ninguna memoria RAM puesto que con los registros del procesador es suficiente.

- Fiabilidad: en función de la responsabilidad que tiene cada dispositivo, deberá cumplir con unos requisitos que den garantías de su buen funcionamiento. Un sistema en un juguete para niños podrá fallar más a menudo que un sistema de control de un misil nuclear, por ejemplo.

- Consumo energético: cualquier dispositivo portátil, como los terminales de telefonía móvil o las computadoras portátiles, estará alimentado por baterías. Interesará, pues, minimizar el consumo energético de manera que se maximice el tiempo de autonomía del dispositivo.

- Esperanza de vida: el tiempo que el sistema deberá continuar funcionando correctamente (de acuerdo con parámetros estadísticos) afecta a muchas decisones en el diseño, desde la selección de los componentes hardware hasta el costo de desarrollo y producción. No hay que descartar tácticas empresariales de dudosa ética como la llamada “obsolescencia programada”.

- Coste: por un lado está el coste de los procesos de diseño y desarrollo de hardware y software, que es fijo y ocurre una única vez. Por otro lado se encuentra el coste de producción del sistema final, que normalmente está en un compromiso con el número de unidades producidas y vendidas. Por ejemplo, no interesará desarrollar los propios componentes hardware para un sistema con un volumen de producción reducido.

Todas estas variables, y otras que también pueden condicionar las características del sistema final, a menudo entran en conflicto entre sí y por lo tanto es necesario llegar a un compromiso entre ellas. Por ejemplo, la esperanza de vida de un dispositivo se puede alargar seleccionando unos componentes más fiables, pero esto a su vez incrementará el costo de producción.

1.3.2 Características de hardware

De manera general, un sistema embebido y una PC comparten una arquitectura semejante, tal y como está representada de forma sencilla en la figura 1.1.

En la parte central se encuentra un microprocesador, un microcontrolador o un procesador digital de señal (DSP) que conforma la unidad central de proceso (CPU), es decir, la que aporta capacidad de cómputo al sistema, pudiendo incluir memoria interna o externa, o en principio cualquier arquitectura específica, según requisitos. Los subsistemas de memoria y de entrada/salida (E/S) se interconectan mediante los buses del sistema (compuestos a su vez por el bus de control, el bus de direcciones y el bus de datos).

El subsistema de entrada acepta datos del exterior para ser procesados mientras que el subsistema de salida transfiere los resultados hacia el exterior. Lo más habitual es que haya varios subsistemas de entrada y varios de salida. A estos subsistemas se les reconoce habitualmente como periféricos de E/S.

El subsistema de memoria almacena las instrucciones que controlan el funcionamiento del sistema. Estas instrucciones forman el programa que ejecuta el sistema. La memoria también almacena varios tipos de datos: datos de entrada que aún no han sido procesados, resultados intermedios del procesado y resultados finales en espera de salida al exterior.

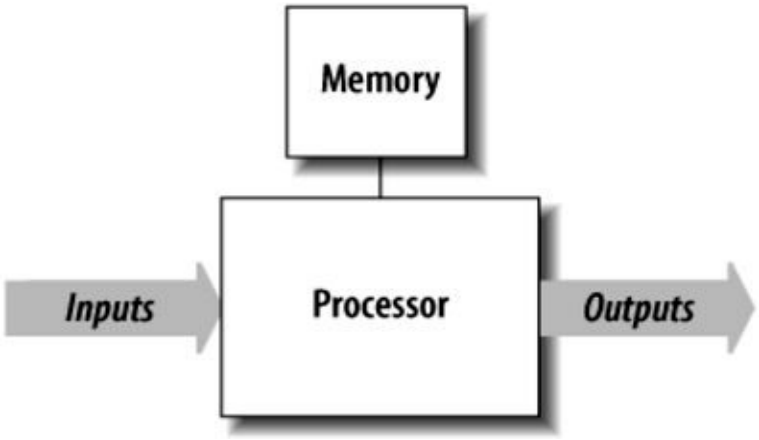

Figura 1.1 *Esquema sencillo de una PC o de un sistema embebido*

No obstante, como ya se ha comentado, las restricciones que presentan los sistemas embebidos hacen que su hardware presente una configuración diferente que las PCs, normalmente teniendo menos recursos que estas últimas. Tómese como ejemplo la PC arbitraria que se representa en la figura 1.2. Tiene una gran memoria principal para albergar al sistema operativo, aplicaciones y datos, y una interfaz para dispositivos de almacenamiento masivo (discos duros y DVD/CD-ROMs). Posee varios dispositivos de E/S para el usuario (teclado, ratón, micrófono, pantalla gráfica y audio) así como de conectividad (periféricos y red). El rápido procesador requerirá de un gestor del sistema para monitorizar la temperatura del núcleo y las tensiones de alimentación, así como para poder generar un *reset*.

Algunos tipos de sistemas embebidos, los de más alto rendimiento, pueden presentar un aspecto semejante a éste. Por ejemplo, si funcionan como *router* o *gateway*, necesitarán una o varias interfaces de red, amplia memoria y alta velocidad de operación. Puede que también requieran algún tipo de interfaz de usuario como parte de su aplicación embebida, y en el caso extremo, simplemente son computadoras convencionales dedicadas a una tarea específica.

Sin embargo, dependiendo de las especificaciones que tengan que cumplir, los sistemas embebidos irán reduciendo los recursos y dispositivos disponibles, pudiendo llegar a la mínima expresión. Los sistemas embebidos más pequeños utilizan microcontroladores como procesador, con la ventaja de que este procesador incorporará gran parte de las funcionalidades de la computadora en un solo chip. En la figura 1.3 se muestra un sistema embebido arbitrario basado en un microcontrolador genérico.

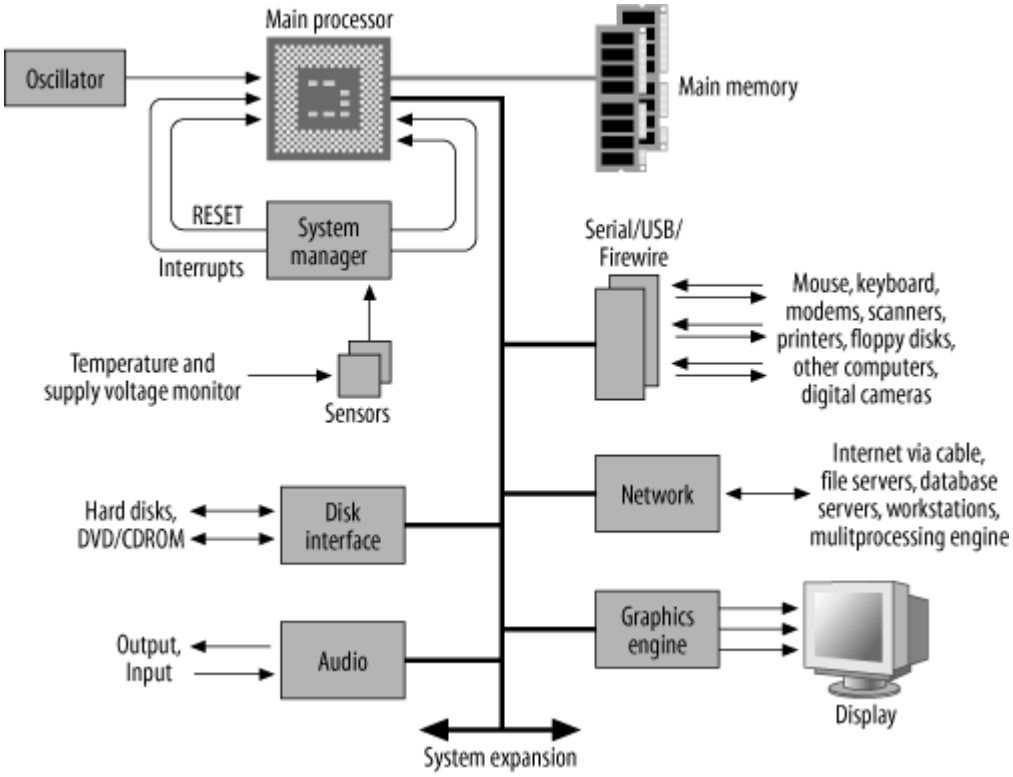

Figura 1.2 *Diagrama de bloques de una computadora genérica*

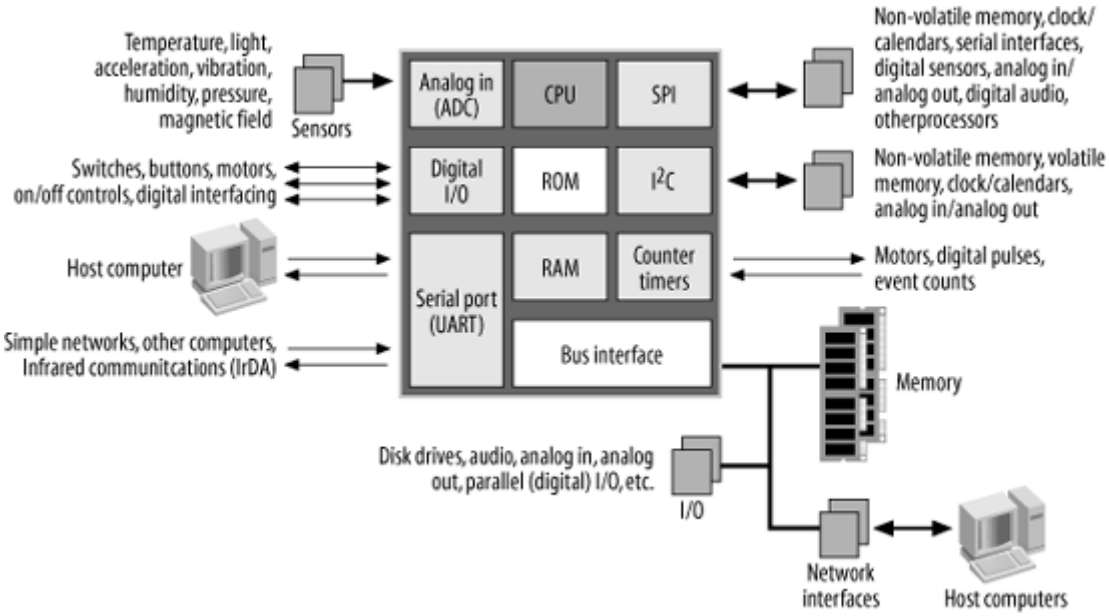

Figura 1.3 *Diagrama de bloques de un sistema embebido arbitrario*

Un microcontrolador es un circuito integrado que incluye una CPU, memoria y circuitos de E/S. Entre los subsistemas de E/S que incluyen los microcontroladores se encuentran los temporizadores, los convertidores analógico a digital (ADC) y digital a analógico (DAC) y los canales de comunicaciones serie. Estos subsistemas de E/S se suelen optimizar para aplicaciones específicas (por ejemplo audio, video, procesos industriales, comunicaciones...) y se implementan dentro del microcontrolador como bloques-subsistemas.

A continuación se presentan los elementos que con más frecuencia están presentes en las arquitecturas embebidas, aunque, como se ha comentado, pueden aparecer o no en un dispositivo en concreto. Estos elementos son muchas veces el objeto de las propias restricciones, siendo los más críticos el procesador y la memoria:

- Microprocesador: es el encargado de realizar las operaciones de cálculo principales del sistema y el que aporta capacidad de cómputo. Ejecuta código (conjuntos de instrucciones) para realizar una determinada tarea y dirige el funcionamiento de los demás elementos que integran el sistema. La potencia de procesamiento se puede medir en términos del número de instrucciones que es capaz de ejecutar por unidad de tiempo, generalmente en MIPS (millones de instrucciones por segundo). Otra característica importante es el ancho de palabra del registro, que puede variar típicamente entre 8 y 64 bits. Las PCs utilizadas hoy en día utilizan ya exclusivamente procesadores de 32 y 64 bits, sin embargo todavía es común en los sistemas embebidos utilizar procesadores más económicos de 8 o 16 bits.

- Memoria RAM o principal: en ella se encuentra almacenado el código de los programas que el sistema puede ejecutar así como los datos. Su característica principal es que debe tener un acceso de lectura y escritura lo más rápido posible para que el microprocesador no pierda tiempo en tareas que no sean meramente de cálculo. Al ser volátil el sistema requiere de un soporte donde se almacenen los datos incluso sin disponer de alimentación o energía. La cantidad de memoria debe ser dimensionada correctamente por el diseñador y también puede afectar a la elección del procesador. En general, el ancho de palabra del registro del procesador establece el límite superior de la cantidad de memoria a la que puede acceder. Por ejemplo, un registro de direcciones de 8 bits sólo puede seleccionar una de entre 256 posiciones de memoria únicas.

- Caché: es una memoria más rápida que la principal en la que se almacenan los datos y el código accedido últimamente. Dado que el sistema realiza microtareas, muchas veces repetitivas, la caché permite ahorrar tiempo ya que no hará falta ir a memoria principal si el dato o la instrucción ya se encuentra en la caché. Dado su alto precio tiene un tamaño muy inferior con respecto a la principal . En el interior del chip del microprocesador se encuentra una pequeña caché, pero normalmente se tiene una mayor en otro chip de la placa madre.

- Disco duro: en él la información no es volátil y además puede conseguir capacidades muy elevadas. A diferencia de la memoria que es de estado sólido éste suele ser magnético. Pero su excesivo tamaño a veces lo hace inviable

para PCs embebidas, con lo que se requieren soluciones como unidades de estado sólido. Otro problema que presentan los dispositivos magnéticos, a la hora de integrarlos en sistemas embebidos, es que llevan partes mecánicas móviles, lo que los hace inviables para entornos donde estos estarán expuestos a ciertas condiciones de vibración. Existen en el mercado varias soluciones de esta clase (DiskOnChip, CompactFlash, IDE Flash Drive...) con capacidades suficientes para la mayoría de sistemas embebidos (desde 2 MiB hasta más de 1 GiB).

- BIOS-ROM (*Basic Input & Output System-Read Only Memory,* memoria de sólo lectura para el sistema básico de entrada y salida) es un chip de memoria específico para albergar un código fijo, necesario para inicializar la computadora y para poner en comunicación los distintos elementos de la placa madre. En muchos sistemas embebidos no existe tal memoria BIOS y el proceso de arranque lo realizan programas instalados en otro tipo de memorias, como por ejemplo Flash, a los que se conoce como *bootloaders*.

- CMOS-RAM: es un chip de memoria de lectura y escritura alimentado con una pila donde se almacena el tipo y ubicación de los dispositivos conectados a la placa madre (disco duro, puertos de entrada y salida, etc.). Además contiene un reloj en permanente funcionamiento que ofrece al sistema la fecha y la hora.

- Memoria flash: es una tecnología de almacenamiento —derivada de la memoria E^2PROM— que permite la lectura/escritura de múltiples posiciones de memoria en la misma operación. Existen dos tecnologías de memoria flash (tipo NAND y y tipo NOR, según el tipo de puertas lógicas utilizadas), cada una con sus características de densidad de almacenamiento, coste, velocidad de lectura/escitura/borrado, fiabilidad, etc.

- Chipset: chip que se encarga de controlar las interrupciones dirigidas al microprocesador, el acceso directo a memoria (DMA) y al bus ISA, además de ofrecer temporizadores y otras funcionalidades. Es frecuente encontrar la CMOS-RAM y el reloj de tiempo real en el interior del chip .

- Periféricos de E/S: los sistemas embebidos pueden tener una gran variedad de subsistemas de entrada/salida, tanto de control como de datos. Dependiendo de su función, pueden disponer de una o varias interfaces para interactuar con el usuario, tales como teclado, monitores, botones, dispositivos táctiles... En otras ocasiones, sin embargo, su tarea se ejecuta en un segundo plano y su existencia puede no ser aparente para el usuario, como por ejemplo el sistema antibloqueo de frenos en un automóvil.

  Uno de los subsistemas de entrada/salida más comunes es el llamado *general-purpose input-oputput* (GPIO) que está basado en la lectura o escritura de valores digitales binarios en ciertos pines del circuito. Se puede utilizar para leer el estado de interruptores, botones u otros dispositivos externos, o bien para activar o desactivar acciones, procesos, etc. Aunque la mayoría de los microcontroladores tienen otros subsistemas de E/S además de GPIO, se suele proporcionar la posibilidad de convertir los otros subsistemas a GPIO si las funcionalidades de aquellos no son requeridas.

Muchos sistemas embebidos necesitan además entradas analógicas, por ejemplo procedentes de sensores de variables físicas, como temperautra, aceleración, presión acústica, intensidad lumínica, campo magnético, etc, para registrarlas, monitorizarlas o procesarlas. En tal caso el sistema deberá disponer también de un conversor ADC para digitalizar las señales.

Algunos sistemas embebidos disponen de puertos serie, que permiten la comunicación bien con una computadora anfitriona o *host,* bien con un modem, con otro sistema embebido o quizás acceder a una sencilla red. Existen formas especializadas de interfaces serie, tales como SPI e I²C, que permiten al procesador tener acceso a dispositivos periféricos tales como memorias externas, chips de reloj para sincronización, sensores digitales o analógicos, otros procesadores... Algunos sistemas embebidos también incluyen interfaces de red como Ethernet, USB o incluso *wireless.*

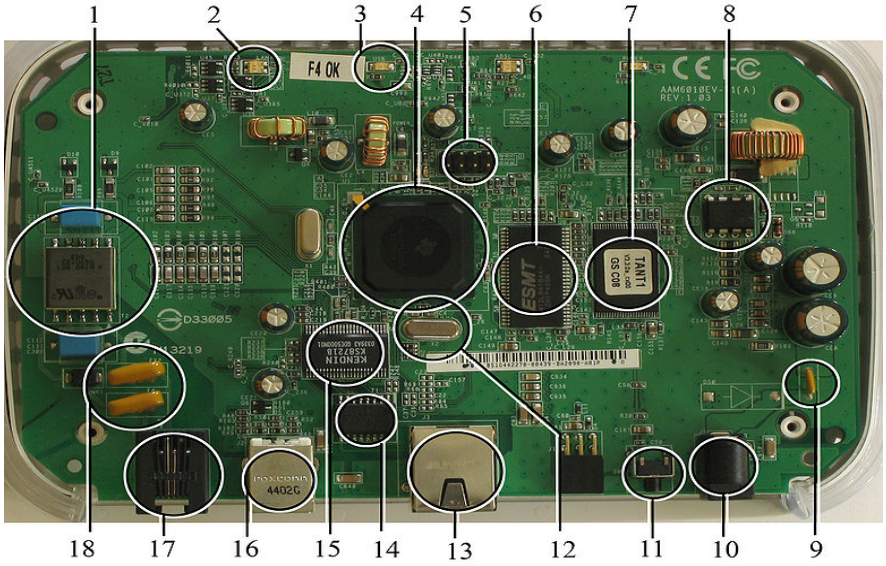

Figura 1.4 *Componentes de un router/modem ADSL Netgear DG632. Las partes marcadas incluyen un microprocesador (4), una memoria RAM (6) y una Flash (7)*

1.3.3 Características de software

Como se vio en la sección anterior, no existe una única arquitectura hardware para todos los dispositivos embebidos, sino que hay mucha variedad y flexibilidad en las configuraciones y cada una está condicionada por sus necesidades particulares. De la misma manera, el software que se ejecuta sobre los dispositivos no es el mismo para todos, sino que depende en gran medida de la aplicación concreta, de la configuración hardware en particular, del número de capas que estén dando soporte por debajo a ese software, etc.

El concepto de software en el ámbito embebido es similar al del de las computadoras convencionales, aunque tiene ciertas peculiaridades que afectan a los programadores. Básicamente, la idea que debería tener en mente una persona con experiencia desarrollando aplicaciones para una PC a la hora de desarrollar software embebido es que su ámbito de actuación va a estar ahora más cerca del hardware. Si tomamos como referencia, por ejemplo, el modelo de capas del sistema operativo osFree que se muestra en la figura 1.5, esto significa que el programador va a trabajar más abajo en el diagrama, con menos capas de abstracción (librerías, interfaces de programación de aplicaciones (APIs), dependencias, etc.) en las que apoyarse. En definitiva, va a programar a más bajo nivel.

Por otra parte, dado que los recursos son más limitados, el código debe hacer uso de ellos de forma especialmente eficiente. Como consecuencia, la programación se vuelve más específica para cada hardware, lo cual hace que las aplicaciones embebidas sean, en general, menos portables que en el mundo de las PCs.

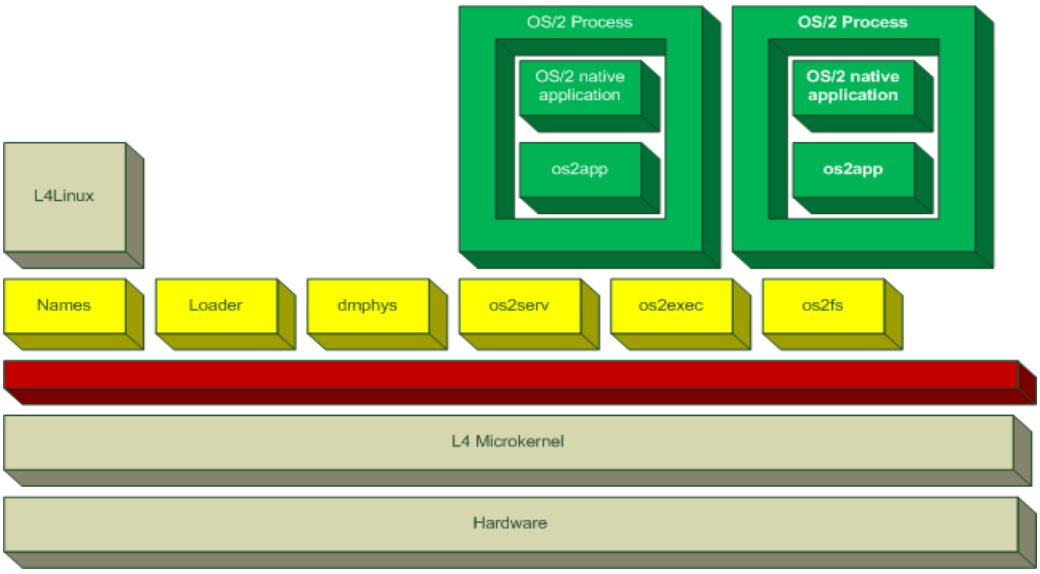

Figura 1.5 *Arquitectura del sistema operativo osFree*

- **Sistemas operativos embebidos**

Se dan casos como el del reloj digital en los que la estructura es tan simple y las tareas a realizar son tan sencillas, que el desarrollo de software conlleva en sí mismo desde el nivel de aplicación hasta el sistema operativo, dando como producto una única imagen ejecutable que contiene todas las tareas para el sistema. Es decir, aquí el código está muy “pegado” al hardware.

En otros casos, en cambio, tales como las aplicaciones para *smartphones* que tan de moda están hoy en día, el software se ejecuta sobre una compleja estructura estratificada de librerías, APIs, dependencias, etc, que a su vez reposa sobre un sistema operativo, el cual toma una arquitectura adecuada al dispositivo. Este sistema operativo funciona de manera semejante al de una PC, gestionando los procesos básicos del sistema pero, naturalmente, estará diseñado para ser más compacto y eficiente, y por contra carecerá de algunas funcionalidades que ofrecen los sistemas operativos mayores.

Entre los dispositivos que cuentan con una estructura que se puede considerar un "sistema operativo", destaca el conjunto de los mencionados *smartphones*, PDAs, *tablets*, teléfonos móviles convencionales y localizadores GPS. Existen en el mercado más de media docena de empresas que están desarrollando sistemas operativos diferentes para este tipo de dispositivos a un nivel muy competitivo entre ellas, presentando diferentes filosofías tecnológicas y estrategias empresariales. En cualquier caso, éste es un sector en auge y con buenas perspectivas de mercado. Esta cuestión se discutirá en mayor profundidad en la siguiente sección.

- **Lenguajes de programación**

Como se ve, la programación embebida se hace a más bajo nivel, pero esto no significa que el programador tenga que descender todo el tiempo al lenguaje ensamblador. El lenguaje utilizado en la mayoría de las aplicaciones embebidas es C. Se trata de un lenguaje simple y relativamente fácil de aprender, que dispone de las estructuras típicas de los lenguajes de alto nivel pero que, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Existen compiladores para prácticamente cualquier procesador, que además suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos. En ocasiones, si la arquitectura así lo permite, es posible hacer una programación orientada a objetos en C++ o C#, aunque la programación embebida es tradicionalmente estructurada. Otros lenguajes de programación que se vienen utilizando, sobre todo con los dispositivos con mayor capacidad para añadir nuevas capas, son Java, Pearl, Python, PHP, Ada...

Sistemas de tiempo real

Una subclase especial dentro de los sistemas embebidos son los sistemas de tiempo real. Para estos sistemas, de entre las especificaciones que deben cumplir, resulta crítico el tiempo de respuesta del sistema. Un sistema de tiempo real debe responder dentro de un intervalo restringido de tiempo a eventos externos mediante la ejecución de la tarea asociada con cada evento. Los sistemas de tiempo real se pueden caracterizar como blandos o duros. Si un sistema de tiempo real blando no cumple con sus restricciones de tiempo, simplemente se degrada el rendimiento del sistema, pero si el sistema es de tiempo real duro y no cumple con sus restricciones de tiempo, el sistema fallará. Este fallo puede tener consecuencias catastróficas, por ejemplo en el caso del sistema de control del vuelo de un avión.

1.4 Estado del arte: historia y situación actual de los sistemas embebidos

Tal y como se definieron los sistemas embebidos en la sección 1.1, resulta imposible que hubieran podido existir antes de 1971, ya que fue en ese año cuando la compañía norteamericana Intel lanzó al mercado el primer microprocesador de la historia, el 4004, diseñado para ser utilizado en una línea de calculadoras producida por la compañía japonesa Busicom. Dos años antes, en 1969, Busicom había solicitado a Intel diseñar y fabricar un conjunto de circuitos integrados para una serie de modelos de calculadoras para negocios, de manera que cada calculadora tuviera su propio circuito específico. Sin embargo, la propuesta de Intel fue diseñar un circuito de propósito general que pudiera ser utilizado en la línea entera de calculadoras. Este procesador de propósito general, el 4004, fue diseñado para leer y ejecutar un conjunto de instrucciones que estaban almacenadas como software en un chip externo de memoria. La idea de Intel era que sería el software lo que porporcionaría a cada calculadora su conjunto único de características.

Aquel microprocesador tuvo un éxito instantáneo y su uso creció progresivamente durante la siguiente década. Las primeras aplicaciones embebidas incluyeron pruebas espaciales no tripuladas, semáforos de tráfico computarizados y sistemas de control para la aviación. En la década de 1980, los sistemas embebidos fueron ganando espacios conforme se instauraba la era de las microcomputadoras, y acabaron llegando a todos los rincones de la vida personal y profesional de los países desarrollados. Como ya se dijo, en estos países, la mayoría de los equipos electrónicos que se encuentran en las viviendas o en los puestos de trabajo hoy en día son sistemas embebidos. En la última década la tendencia hacia la reducción de tamaño de las computadoras personales y su transición hacia otro tipo de dispositivos con nuevas funcionalidades como los *smartphones* y demás, ha hecho aumentar aún más la repercusión de este tipo de sistemas.

Actualmente, ante los retos que presenta la globalización y la fuerte presión de los mercados emergentes, casi todos los sectores están haciendo esfuerzos que les permitan aumentar sus cuotas de competitividad. La gran aplicabilidad de los sistemas embebidos en cualquier ámbito sectorial, así como el valor añadido que aportan los mismos a los productos que los contienen, hace que el desarrollo de estos sistemas sea un área estratégica preferente para muchas empresas que buscan precisamente este aumento de su competitividad. Así, los sistemas embebidos van a jugar un papel vital en nuestra sociedad y se supone revolucionarán los sectores de actividad, como son el sector médico, el de medios de transporte o el de automatización industrial, entre otros. Como dato que refleja la importancia que este tipo de sistemas ha adquirido, se estima que el volumen de mercado global en 2010 -en plena crisis económica mundial- rondó los 200 billones de euros.

A continuación se analiza brevemente la implantación de este tipo de sistemas en los estados de España y Cuba, es decir, los países donde se llevó a cabo este trabajo.

1.4.1 En España

De acuerdo con un estudio de la Fundación OPTI y la Fundación ASCAMM, en 2009 Europa era el máximo representante en este campo a nivel mundial, donde se esperaba que para el año 2010 , el porcentaje de inversión en investigación y desarrollo (I+D) en sistemas embebidos, sobre el total de la inversión en I+D, fuese del 14%. Aun así, esta posición ventajosa podría perderse a favor de los Estados Unidos o de algunos países emergentes, sobre todo asiáticos. Los Estados Unidos tienden a utilizar los resultados de los sistemas embebidos obtenidos en las aplicaciones militares e industriales y los países asiáticos poseen un amplio mercado nacional y el *know-how* tecnológico en fabricación que ponen en peligro la posición de liderazgo europea. Europa, por otro lado, posee una elevada cualificación profesional en este ámbito, unos mercados desarrollados y una buena infraestructura.

Una parte de esta infraestructura la conforman las plataformas europeas relacionadas con los sistemas embebidos. Las más destacadas son la Plataforma ARTEMIS y la Plataforma ENIAC. Mediante estas plataformas se ha conseguido crear una gran red de contactos que facilita la transferencia de conocimiento y potencia la colaboración a nivel europeo. Su importancia es tal que llega a influir en la toma de decisiones de las líneas de investigación en el marco europeo.

Dadas estas condiciones favorables, la industria y la investigación españolas pueden tener un papel muy representativo en el campo de los sistemas embebidos. Para ello será necesario identificar y tener en cuenta cuáles son las tendencias emergentes que probablemente serán de mayor relevancia en este campo en los próximos años.

1.4.2 En Cuba

En Cuba los sistemas embebidos no han tenido aún un impacto tan espectacular en la vida doméstica y laboral como en otras partes del mundo, si bien están siendo implantados en mayor o menor medida a nivel industrial, militar y científico. Existen líneas de investigación sobre sistemas embebidos, al menos, en el Instituto Superior Politécnico José Antonio Echeverría CUJAE y en la Universidad de Ciencias Informáticas (UCI), ambos en la provincia de Ciudad de La Habana. En esta última universidad se desarrolló (y continúa desarrollándose) el proyecto Nova, una distribución cubana de GNU/Linux que surgió con la idea de funcionar en computadoras con pocas prestaciones de hardware y que, a diferencia de otras distribuciones de mayor estandaridad, se especializa más en personalizaciones a la medida, con lo que presenta grandes ventajas para su uso en dispositivos embebidos.

En la Universidad de Pinar del Río Hermanos Saíz Montes de Oca, donde se realizó la mayor parte de este trabajo, existe un interés creciente por este tipo de sistemas. El departamento de Ingeniería Electrónica y de Telecomunicaciones, que cuenta con una larga tradición en el empleo de microcontroladores PIC, está trabajando actualmente con una placa Mini2440 de FriendlyARM. Por otra parte, el Grupo de Investigaciones de Diagnóstico Avanzado de Maquinarias (GIDAM) ha conseguido muy buenos resultados embebiendo un DSP y una pequeña computadora en una FPGA, así como un sistema operativo basado en un kernel µClinux, para aplicaciones en el ámbito del diagnóstico industrial y la ingeniería biomédica.

El campo de los sistemas embebidos puede llegar a tener una importancia estratégica para los países de América Latina, particularmente para Cuba. Estos sistemas aportan valor añadido a los productos y, cada vez más, son los responsables de las mejoras introducidas en términos de innovación y competitividad. Por ello es un campo que puede jugar un papel importante en tiempos de creación y consolidación de un tejido industrial cohesionado y con perspectivas de desarrollo.

2. LA PLACA DE DESARROLLO MINI2440

2.1 Introducción

En este capítulo se da una descripción del dispositivo embebido Mini2440, que constituye la herramienta central de este proyecto, ya que sobre él ha de ejecutarse la aplicación cuyo diseño es el objeto de este trabajo.

El Mini2440 es un dispositivo fabricado en China y distribuido por compañías como FriendlyARM, Industrial ARMworks, Andahammer, Hiteg o ThaiEasyElec. A rasgos generales, se trata de un sistema de computación versátil y de propósito múltiple, con un tamaño de bolsillo, pero también con notables limitaciones de recursos con respecto a una PC actual. En el hipotético eje que tiene en un extremo a las PCs y en el otro a los dispositivos hardware-monotarea, el Mini2440 se encontraría en una zona intermedia, en el semieje más cercano a las PCs, compartiendo espacio con *smartphones*, *tablets* y similares.

El Mini2440 está concebido sobre todo como una placa de desarrollo, es decir, con la finalidad de que los programadores desarrollen y prueben en ella sus aplicaciones. Estas aplicaciones pueden ser de carácter muy diferente entre sí, ya que el abanico de posibilidades que nos brinda el dispositivo es muy amplio; se le puede conectar una cámara web, un módulo GPS o una tarjeta de televisión, por poner algunos ejemplos. Además de su versatilidad, otra características como su robustez y su precio reducido hacen que sea de gran utilidad en el campo de la docencia y de la investigación.

Se comercializa en distintas versiones, presentando cada una ligeras diferencias en cuanto a su configuración y prestaciones. Por ejemplo, a través de FriendlyARM se puede adquirir sin *display* gráfico, o con un *display* LCD táctil fabricado bien por Sony o bien por Toppoly, en versiones de 3.5” o de 7”. La memoria NAND Flash puede ser de 64, 128 o 256 MB. El *kit* concreto sobre el que se trabajó en este proyecto cuenta con el *display* gráfico LCD táctil de 3.5” 240x320 de Toppoly, memoria NAND Flash de 128 MB e incluía con su compra un adaptador de alimentación eléctrica, cables de conexión serie RS232 y USB, un cable de red Ethernet *crossover,* un adaptador JTAG y un DVD con diverso material de consulta así como paquetes de software, tales como los sistemas operativos Qtopia y WinCE 5.0 y aplicaciones de test para los subsistemas. Fue adquirido por aproximadamente 90€ en http://www.friendlyarm.net/. Un *pack* similar se muestra en la figura 2.1.

Es de destacar, en el aspecto negativo, la falta de manuales y material explicativo para esta placa, que sean accesibles para principiantes en el mundo de la programación embebida y que estén debidamente estructurados y traducidos al menos al inglés. La

información contenida en el DVD es escasa, dispersa y en muchos casos se encuentra sólo disponible en chino. Afortunadamente, la comunidad de usuarios en internet de ésta y otras placas de desarrollo hace un gran aporte en este sentido, si bien durante la realización de este trabajo se dispuso de una conexión a internet muy limitada en cuanto a ancho de banda y tiempo de conexión, la cual no permitió hacer uso de todo el potencial de dicho aporte (foros, blogs, sitios ftp, etc.).

Figura 2.1 *El kit Mini2440 de FriendlyARM*

2.2 Hardware

El Mini2440 está construido sobre un chip de 10 x 10 cm, y cuenta con una gran variedad de interfaces y sistemas de almacenamiento que están montados sobre la placa, tal y como se muestra en la figura 2.2.

Éste es un listado de todos los componentes hardware de la placa:

- Microprocesador:
  - S3C2440A de arquitectura ARM920T de 32 bits
  - Frecuencia 405 MHz (Máx. 533 Mhz)
  - 200 MIPS efectivas @ 180 MHz

- Memoria RAM:
  - SDRAM 64 MB montada sobre la placa
  - Bus de datos de 32 bits
  - Frecuencia de reloj de hasta 100 Mhz

- Memoria Flash:
  - NAND Flash de 128 MB (la configuración de la figura 2.2 tiene 64 MB)
  - NOR Flash de 2 MB con BIOS instalada

- Interfaz LCD:
  - Interfaz táctil resistiva de 4 canales conectada al ADC de 10 bits de resolución
  - Soporte para LCD STN escala de grises de 4 y 16 niveles, 256 y 4096 colores
  - Soporte para LCD TFT escala de grises de 4 y 16 niveles, 256, 64k colores y *True Color*
  - Tamaño de pantalla 3.5". Existen versiones de hasta 12.1"
  - Resolución máxima de pantalla de 1024 x 768 píxeles
  - Configuración estándar para el LCD táctil NEC 240 x 320 / 3.5" TFT *True Color*
  - Conectada a una interfaz de alimentación *on-board* de 12V, para la alimentación del módulo *backlight* LCD TFT 12V CCFL

- Otras interfaces y accesorios externos:
  - Una interfaz 10/100M Ethernet RJ-45 (chip de red DM9000)
  - 3 puertos serie, uno configurado para RS-232 (COM0)
  - Un USB host
  - Una interfaz USB de tipo Slave B
  - Una interfaz para tarjetas de almacenamiento SD, sin límite de tamaño
  - Una interfaz de salida de audio stereo
  - Un micrófono integrado
  - Una entrada de micrófono
  - Una interfaz JTAG de 10 pines de 2.0mm de pitch
  - 4 LEDs de usuario
  - 6 botones de usuario (*user buttons*), con conexión a conectores GPIO y de usuario de 8 pines
  - Un botón de reinicio (*reset*), conectado a un chip de *reset*.
  - Un interruptor de selección de arranque desde Flash NAND/NOR
  - Un potenciómetro (resistencia variable) para test del ADC (*A/D test*)
  - Un timbre PWM (*buzzer*)
  - Un chip de bus IIC AT24C08 para datos de configuración o prueba, abarca 256 bytes.
  - Una interfaz de cámara de 20 pines de 2.0 mm de pitch
  - Una interfaz de alimentación (5V), con interruptor de encendido (*power switch*) y luz indicadora

- Fuente de reloj del sistema:
  - De cristal pasivo
  - Reloj interno de tiempo real (con batería de litio de respaldo)

- Interfaz de expansión:
  - Una interfaz GPIO de 34 pines de 2.0 mm de pitch
  - Una interfaz de bus del sistema de 40 pines de 2.0 mm de pitch

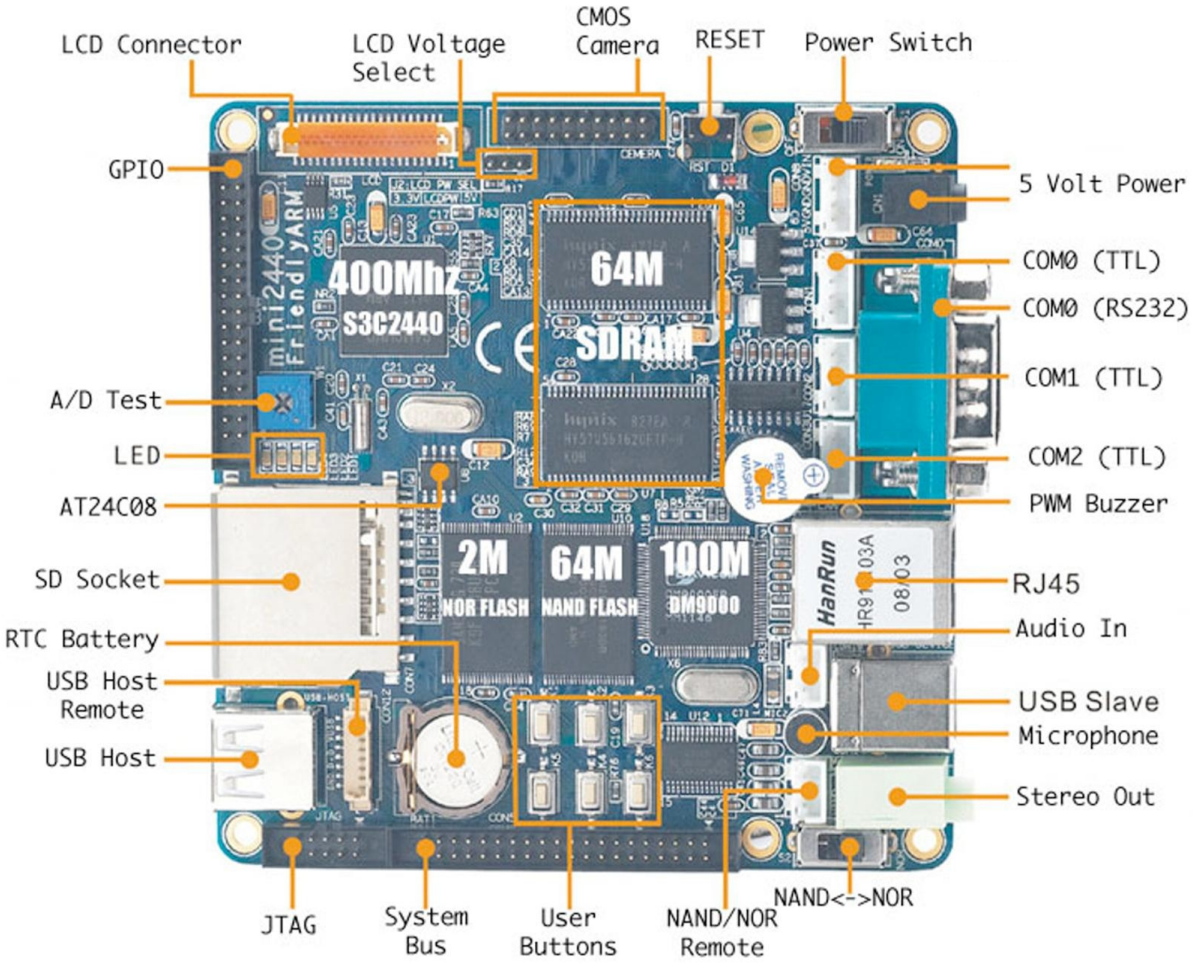

Figura 2.2 *Configuración de los componentes hardware en una placa Mini2440*

2.2.1 Microprocesador ARM S3C2440A

- **ARM**

ARM (*Advanced RISC Machines*) es la denominación de un conjunto de microprocesadores con una arquitectura y un juego de instrucciones RISC (*reduced instruction set computer*) de 32 bits que fueron diseñados por Acorn Computers y desarrollados por ARM Holdings Ltd, ambas compañías británicas.

Fue originalmente concebida por Acorn Computers como procesador para PCs de sobremesa, un mercado dominado por la familia x86 que era usada por los computadores IBM PC. Sin embargo, la relativa sencillez de los procesadores ARM los hacían apropiados para aplicaciones de bajo consumo eléctrico. Esto los ha hecho liderar el mercado de la electrónica móvil y de los dispositivos embebidos como microprocesadores y microcontroladores pequeños y asequibles.

Atendiendo a datos de producción, la arquitectura ARM es el repertorio de instrucciones de 32 bits más extendido actualmente. Hasta el año 2007 el 98 % de los más de mil millones de teléfonos móviles vendidos anualmente usaban al menos un procesador ARM. En 2009 aproximadamente el 90 % de los procesadores RISC de 32 bits empleados en dispositivos empotrados de 32 bits eran ARM. Estos procesadores son ampliamente usados en electrónica de consumo, incluyendo PDAs, iPods y otros reproductores, videoconsolas portátiles, calculadoras y periféricos de computador como discos duros y routers.

La arquitectura ARM es licenciable. Algunas compañías que son o han sido licenciadas incluyen: Alcatel, Atmel, Broadcom, Cirrus Logic, Digital Equipment Corporation, Freescale, Intel (a través de DEC), LG, Marvell Technology Group, NEC, NVIDIA, NXP (previamente Philips), Oki, Qualcomm, Samsung, Sharp, ST Microelectronics, Symbios Logic, Texas Instruments, VLSI Technology, Yamaha y ZiiLABS.

- **S3C2440A**

El núcleo del Mini2440 es el microprocesador S3C2440A, de arquitectura ARM920T de 32 bits, fabricado por la empresa surcoreana Samsung.

La arquitectura ARM920T está compuesta por un núcleo ARM9TDMI más una unidad de gestión de memoria (MMU, *memory management unit*) y memoria caché. El núcleo es un dispositivo de arquitectura Harvard, implementado utilizando un *pipeline* de cinco etapas: *fetch, decode, execute, memory,* y *write*. Las cachés de instrucciones y datos tienen 16 KB cada una, con una longitud de 8 palabras. El procesador implementa una arquitectura ARM enriquecida v4 MMU para proporcionar traducción y para permitir chequeos de permiso para las direcciones de datos e instrucciones.

La interfaz del ARM920T con el resto del sistema se hace sobre buses unificados de datos y direcciones. Esta interfaz habilita la implementación de un esquema de bus tipo *Advanced Microcontroller Bus Architecture* (AMBA), *Advanced System Bus* (ASB) o *Advanced High-performance Bus* (AHB). El ARM920T además puede soportar una macrocelda ETM (*Embedded Trace Macrocell*) para el rastreo en tiempo real de datos e instrucciones.

Tipos de datos

Los tipos de datos soportados por el S3C2440A son:

- *byte* 8 bits
- *halfword* 16 bits
- *word* 32 bits
- *double word* 64 bits

Modos de ejecución

El microprocesador S3C2440A soporta los siete modos de ejecución que se muestran en la tabla 2.1:

Modo de ejecución	Número de modo	Descripción
User (usr)	0b10000	Modo de ejecución normal
FIQ (fiq)	0b10001	Interrupciones rápidas
IRQ (irq)	0b10010	Interrupciones de propósito general
Supervisor (svc)	0b10011	Modo protegido para el sistema operativo
Abort (abt)	0b10111	Permite implementar memoria virtual y/o protección de memoria
Undefined (und)	0b11011	Soporte para emulación software de coprocesadores
System(sys)	0b11111	Ejecución de tareas privilegiadas del sistema operativo

Tabla 2.1 *Modos de ejecución en la arquitectura ARM920T*

El cambio de modo puede ser producido por software, a causa de una interrupción externa o por el procesamiento de una excepción.

- La mayoría de las aplicaciones se ejecutan en modo ***User***. En este modo el programa en ejecución no puede acceder a ciertos recursos (memoria, coprocesadores, periféricos...) ni cambiar de modo, a menos que tenga lugar una excepción (ésta se produce mediante la instrucción SWI). Esto permite al sistema operativo controlar el uso de los recursos.

- Todos los modos excepto el modo *User* son modos privilegiados. Tienen total acceso a los recursos del sistema y pueden cambiar el modo de ejecución libremente. Cinco de ellos son los modos de excepción:

  - **FIQ**
  - **IRQ**
  - ***Supervisor***
  - ***Abort***
  - ***Undefined***

  Se accede automáticamente a estos modos cuando se produce una excepción. Cada uno de ellos posee una serie de registros adicionales para evitar corromper el estado del modo User cuando se produce la excepción.

- El modo restante es el modo ***System***, al cual no se accede mediante una excepción y emplea los mismos registros que el modo User. Al ser un modo privilegiado carece de las restricciones del modo *User.* Su uso está indicado para tareas del sistema operativo que necesitan acceder a los recursos del sistema, pero necesitan evitar usar los registros adicionales asociados a los modos de excepción. De este modo se asegura que el estado de la tarea no se altera al ocurrir una excepción.

Registros

En la arquitectura ARM920T, el procesador dispone de 37 registros de 32 bits, de los cuales:

- 6 son registros de estado, aunque no se usan todos los bits.
- 31 son de propósito general, incluyendo el contador de programa. De ellos 16 son visibles en todo momento. El resto se emplean para acelerar el procesamiento de excepciones.

Este banco principal de 16 registros (R0 a R15) puede ser direccionado por el código sin necesidad de privilegios, o sea, en modo *User*. Los registros de este banco, tal y como se muestra en la figura 2.3, están dispuestos en bancos parcialmente solapados, siendo el modo de ejecución actual el que controla qué banco está actualmente visible. Estos registros pueden clasificarse en función de la forma en que se solapan y en sus propósitos específicos:

- **Registros no solapados**, de **R0 a R7**: no se solapan, esto significa que cada uno de ellos se refiere al mismo registro físico de 32 bits en todos los modos de ejecución. Son registros de propósito general puros, sin usos especiales impuestos por la arquitectura.

- **Registros solapados**, de **R8 a R12** : el registro físico al que se refiere cada uno de ellos depende el modo de ejecución actual. Los registros R8 a R12 disponen de dos bancos registros físicos cada uno. Uno de los bancos está reservado para el modo FIQ y el otro es común para el resto de modos. Estos registros no tienen un cometido especial en la arquitectura, sin embargo, para interrupciones que pueden ser tratadas solamente con los registros R8 a R12, la existencia de un banco aislado para el modo FIQ permite un procesamiento muy rápido de las interrupciones.

- El registro **R13** es utilizado por el software como **puntero de pila** (*stack pointer* o **SP**).

- El registro **R14** es el **registro de enlace** (*link register* o **LR**). Tiene dos usos especiales en la arquitectura:
  – Dentro de cada modo, LR contiene la dirección de retorno de la subrutina. El retorno de la subrutina se realiza cargando LR en el contador de programa.
  – Cuando tiene lugar una excepción el LR del modo de ejecución apropiado se carga con la dirección de retorno de la excepción (más un *offset*, dependiendo de la excepción). El retorno se realiza de forma similar al retorno de la subrutina, teniendo en cuenta que es preciso restaurar completamente el estado del programa que se estaba ejecutando.

- El registro **R15** es el **contador de programa** o *program counter* (PC). Contiene la dirección de la siguiente instrucción que se va ejecutar. Lógicamente, las instrucciones tienen ciertas limitaciones al acceder a él. Leer el PC se suele emplear para direccionar instrucciones y datos cercanos independientes de la posición dentro de un programa. Escribir en el PC supone saltar a la dirección escrita en él.

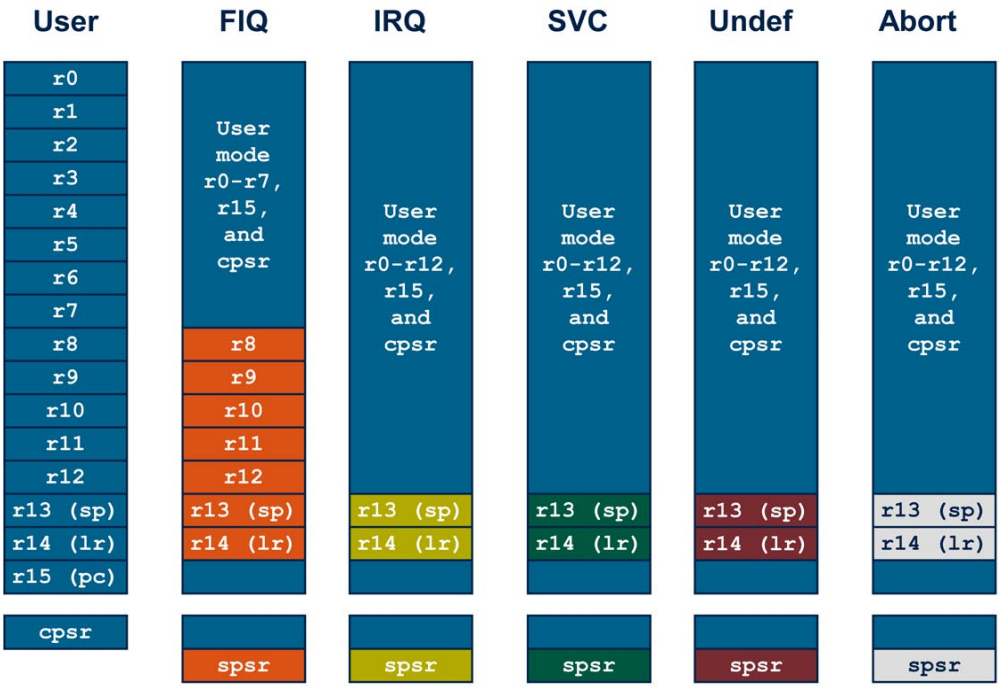

Figura 2.3 *Esquema de los registros del microprocesador S3C2440A*

De entre el resto de registros que no son siempre visibles, destacan los **registros de estado de programa.** El *current program status register* (**CPSR**) es accesible en todos los modos de ejecución. Contiene los bits de condición, el bit para habilitar/inhabilitar las interrupciones, el modo de ejecución y otra información de estado y control. Cada modo de excepción dispone, además, del *saved program status register* (**SPSR**), que contiene el CPSR del programa interrumpido. El formato del CPSR y del SPSR se muestra en la figura 2.4, y a continuación se explican los bits que lo conforman:

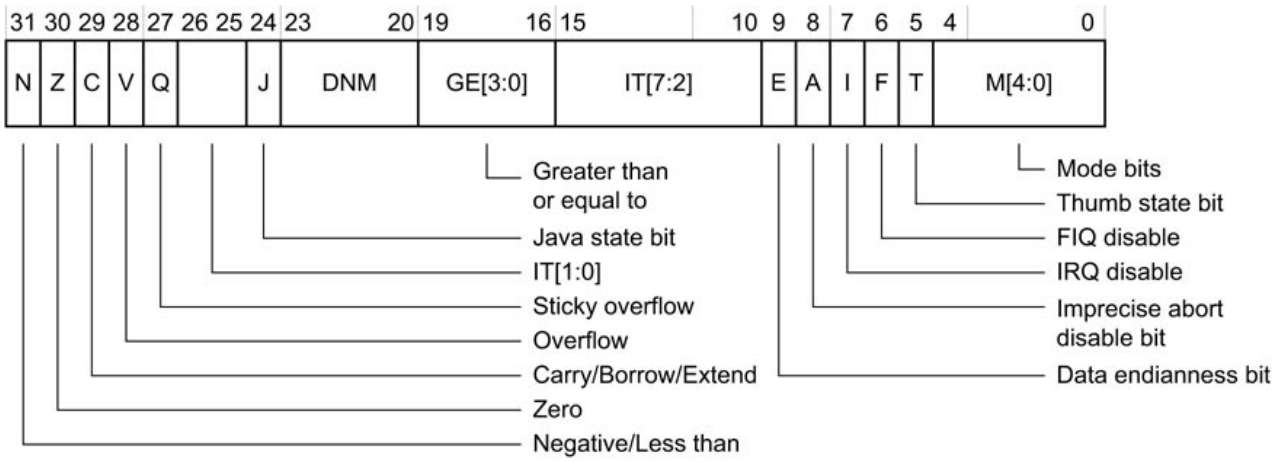

Figura 2.4 *Esquema del registro CPSR/SPSR del S3C2440A*

- **Los bits de condición:** los bits N, Z, C y V (negativo, cero, arrastre y desbordamiento) son los bits de condición o *flags*. Se comprueban para saber si una instrucción condicional debe ser ejecutada.
- **El bit Q:** se usa para indicar que se ha producido un desbordamiento y/o saturación en ciertas instrucciones DSP.
- **Los bits GE|3:0|:** se usan para comprobar relaciones de igualdad y de orden en instrucciones SIMD .
- **El bit E:** controla el *endianess* del manejo de datos
- **Los bits de inhabilitación de interrupciones :** el bit A deshabilita los Data Aborts imprecisos cuando vale 1. El bit I deshabilita las interrupciones IRQ cuando vale 1, mientras que el bit I hace lo propio con las interrupciones IRQ .
- **Los bits de modo:** indican el modo de ejecución del núcleo.
- **Los bits T y J:** se utilizan para seleccionar el juego de instrucciones.

El resto de bits están reservados para futuras expansiones. Es recomendable que el código se escriba de modo que esos bits no sean modificados para prevenir efectos indeseados al ejecutar ese código en versiones futuras de la arquitectura.

Excepciones

La arquitectura ARM920T soporta siete tipos de excepción, cada una de los cuales se procesa mediante un modo privilegiado de ejecución, como se muestra en la tabla 2.2. Los siete tipos son:

- *Reset,* reinicio
- Intento de ejecutar una instrucción indefinida
- Interrupciones software (SWI), pueden usarse para implementar llamadas al sistema operativo
- *Prefetch abort*, error al buscar una instrucción en memoria
- *Data abort*, error al acceder a datos en memoria
- IRQ, interrupciones normales
- FIQ, interrupciones rápidas

Las excepciones son generadas por fuentes internas o externas para que el procesador trate un evento, como una interrupción externa o el intento de ejecutar una instrucción indefinida. Pueden producirse múltiples excepciones al mismo tiempo.

Cuando ocurre una excepción, se fuerza la ejecución de código desde unas posiciones de memoria fijas, denominadas vectores de excepcióny algunos registros comunes son reemplazados por otros específicos del modo.

Todos los modos de excepción disponen de versiones privadas de R13 y R14. El modo FIQ, además, dispone de más registros privados adicionales para un procesamiento rápido de las interrupciones. Normalmente el estado del procesador (el contenido de CPSR y el PC) es guardado en el momento que se produce la excepción para poder recuperar el programa original, una vez el tratamiento de la excepción ha concluido.

Cuando se entra en el manejador de la excepción se guarda en R14 la dirección de retorno. Esto sirve para retornar una vez la excepción ha sido tratada y para conocer la dirección de la instrucción causante de la excepción. El registro R13 es privado en cada modo para permitir punteros de pila independientes. Los registros R8-R12 en el modo FIQ son privados para evitar tener que salvarlos y restaurarlos. *Reset* comparte el modo privilegiado con SWI.

Tipo de excepción	Modo	Dirección Normal	Dirección *High Vector*
Reset	Supervisor	0x00000000	0xFFFF0000
Undefined instructions	Undefined	0x00000004	0xFFFF0004
Software interrupt (SWI)	Supervisor	0x00000008	0xFFFF0008
Prefetch Abort (instruction fetch memory abort)	Abort	0x0000000C	0xFFFF000C
Data Abort (data access memory abort)	Abort	0x00000010	0xFFFF0010
IRQ (interrupt)	IRQ	0x00000018	0xFFFF0018
FIQ (fast interrupt)	FIQ	0x0000001C	0xFFFF001C

Tabla 2.2 *Excepciones en el S3C2440A*

2.2.2 Otros subsistemas importantes

- **Sistemas de almacenamiento Flash**

El Mini2440 tiene 2 memorias Flash diferentes. Una es la NOR Flash de 2MB, modelo SST39VF1601, y la otra es la NAND Flash de 128 MB (en la versión empleada en este proyecto), modelo K9F1208. El procesador permite el arranque desde cualquiera de las dos, seleccionando el modo de arranque a través del interruptor S2.

Normalmente se arrancará desde la NAND Flash, donde estará alojado el sistema operativo y su propio programa bootloader. La NOR Flash contendrá solamente otro bootloader. El arranque desde la NOR se hará para transferir una imagen de sistema operativo a la NAND Flash, o para permitir tareas de recuperación, si por ejemplo el bootloader de la NAND estuviera dañado.

- **Interfaz para el display gráfico LCD táctil**

El chip de interfaz para el display LCD, modelo LCD41P, tiene un conector blanco de 41 pines de 0.5 mm de pitch, que contiene algunas señales de control típicas de los LCDs (reloj, activación, campo escaneado, etc.) y la señal de salida RGB de 8:8:8, que puede soportar un máximo de 16 millones de colores. La iluminación de fondo (*backlight*) es dada a través de LEDs y se puede apagar y encender (y ser regulada su intensidad) a través de GPB1. Los pines 37, 38, 39 y 40 del son utilizados para la interfaz táctil de 4 canales y están conectados directamente a ella. Como todas las interfaces LCD actuales, utiliza una alimentación de 5V.

- **Interfaz GPIO**

El conector CON4 del Mini2440 tiene una interfaz para GPIO con 34 pines de 2.0 mm de pitch. Además de los pines de GPIO, en CON4 se encuentran otros pines para la CPU, como AIN0-AIN3, CLKOUT, interfaces SPI e I2C, GPB0 y GPB1.

- **Conversor analógico-digital (ADC)**

El Mini2440 cuenta con un conversor ADC, modelo UDA1341TS, de 8 canales y 10 bits de resolución, del tipo muestreo y retención (*sample and hold*). La frecuencia de muestreo por defecto es 44.1 kHz. Cuando se está usando el display LCD táctil, 4 canales se utilizan para la conversión de las señales táctiles de entrada. Los otros 4 son de propósito general y son accesibles a través del conector CON4, es decir, el de GPIO. Los pines 5, 6, 7 y 8 de este conector se corresponden con AIN0, AIN1, AIN2 y AIN3, respectivamente. El circuito impreso (PCB) de la placa salida de fábrica tiene AIN0 conectado a un potenciómetro para permitir testar el ADC, como se muestra en la figura 3.5. Si se desea utilizar AIN0 como GPIO, se puede retirar el potenciómetro o cortar la pista que se encuentra en la parte de abajo del PCB con bastante espacio libre a su alrededor.

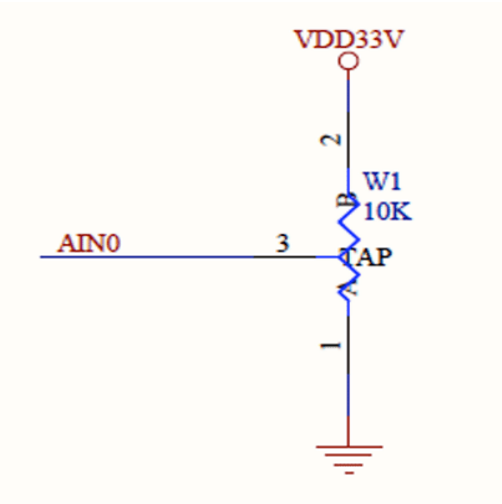

Figura 2.5 *El pin AIN0 del ADC conectado al potenciómetro*

- **Interfaz de red Ethernet**

La placa Mini2440 posee un chip DM9000 10/100M LAN con transformadores de red y un conector RJ-45. Se puede utilizar un cable Ethernet Cat5E para conectar la placa a un PC, router, switch u otros dispositivos. Las direcciones MAC e IP por defecto son idénticas en todos los dispositivos Mini2440 y pueden ser configuradas por software.

- **Sistema de alimentación**

El Mini2440 requiere una alimentación externa de 5V, que se le suministra a través de CON8. Mediante tres reguladores, se generan tensiones de alimentación internas de 3.3V, 1.8V y 1.25V. El interruptor S1 controla el encendido y apagado, no se puede hacer esto mediante software.

- **Reinicio del sistema (reset)**

El Mini2440 dispone de un chip de *reset* modelo MAX811, que proporciona un reinicio de la CPU limpio y de bajo nivel cuando se pulsa el botón RESET.

2.3 Software

- **Sistemas operativos**

Debido a lo complejidad de la configuración hardware de esta placa, está diseñada para ser gobernada por un sistema operativo que gestione los procesos básicos del sistema. El sistema tiene la suficiente capacidad de procesamiento y memoria como para albergar y ejecutar un sistema operativo multiusuario/multitarea con una interfaz gráfica de usuario (GUI, *graphic user interface*) completa.

Esto significa que que el desarrollador deberá en primer lugar instalar un sistema operativo previamente compilado para esta arquitectura, junto con los controladores o *drivers* de los subsistemas. Luego sobre ello programará sus aplicaciones, haciendo uso de las plataformas y librerías disponibles que le puedan dar soporte.

Existen varias alternativas para ser usadas como sistema operativo, teniendo cada una sus particularidades y quedando la elección a libertad del usuario. Algunas de las posibilidades son:

- Linux 2.6.x + Qtopia (Linux)
- Embedian (Linux)
- Android (Linux)
- Ångström (Linux)
- Windows CE 5.0

- Windows CE 6.0
- uCos (sistema operativo para aplicaciones de tiempo real)

En este trabajo se escogió el sistema operativo formado por el kernel de Linux 2.6.29 junto con la plataforma Qtopia, que es la configuración que trae por defecto la placa. Esta configuración software se tratará en profundidad en el siguiente capítulo.

- **Bootloaders**

El Mini2440 no dispone de un chip BIOS-ROM, por tanto necesita al menos un programa bootloader instalado en la memoria NAND Flash, que le permita inicializar el hardware, proporcionándole los parámetros de arranque (especialmente al controlador de memoria) y “levantar” el sistema operativo.

Otra función del bootloader es interactuar con el usuario a través de una consola de comandos para transferir nuevas imágenes binarias a la RAM via Ethernet o puerto serie, copiar imágenes binarias desde la RAM a la Flash, recuperar el sistema operativo, etc. Para ello es necesario tener un bootloader instalado también en la memoria NOR Flash.

En este aspecto también existe una gran variedad de posibilidades. El Mini2440 trae por defecto en ambas Flash el bootloader Supervivi, que es específico para ARM, pero existen otros bootloaders universales como U-boot, RedBoot o uMon, que también pueden usarse en la placa. Por sencillez, aquí también se optó por mantener la configuración de fábrica.

3. HERRAMIENTAS SOFTWARE EMPLEADAS

3.1 Introducción

En este capítulo se describen las principales herramientas software utilizadas en este trabajo, que fueron: el compilador cruzado ARM GCC, el kernel de Linux, la biblioteca Qt y la plataforma Qtopia. Estas herramientas fueron elegidas con el criterio de que los programas empleados fueran libres. Por ello previamente, en la primera sección del capítulo, se definirá el concepto de software libre y se explicarán las ventajas que aporta.

3.2 Software libre

El concepto de software libre surgió en la primera mitad de la década de 1980, cuando el programador estadounidense Richard Stallman inició el proyecto GNU, con la ambición de desarrollar un sistema operativo y en general herramientas software que concedieran a los usuarios libertad para usar, copiar, redistribuir, entender y modificar dichas herramientas. Con el fin de promover el desarrollo y el uso de este tipo de software, Stallman fundó asimismo la *Free Software Foundation* (FSF, Fundación para el Software Libre).

De acuerdo con la definición dada por la FSF, software libre es aquél que garantiza a sus usuarios las cuatro libertades[3] que se describen en la tabla 3.1:

Libertad	Descripción
0	Libertad de usar el programa, con cualquier propósito
1	Libertad de estudiar cómo funciona el programa y de modificarlo para adaptarlo a las necesidades propias de cada usuario
2	Libertad de distribuir copias del programa, de modo que toda la comunidad de usuarios se beneficie
3	Libertad de mejorar el programa y hacer públicas esas mejoras, de modo que toda la comunidad de ususarios se beneficie

Tabla 3.1 *Libertades del software libre*

3 Nótese que para hacer uso de las libertades 1 y 3 es necesario que el usuario tenga acceso al código fuente de los programas, si bien ésta no es una condición suficiente para que un software sea considerado “libre”

El software que no garantiza alguna de estas cuatro libertades, es decir, aquél que presenta limitaciones con respecto a su uso, entendimiento, modificación o distribución, se conoce como software privativo (también llamado software propietario, no libre o de código cerrado).

No obstante, la FSF no es la única organización involucrada en la promoción y la regulación del software desarrollado y distribuido libremente, ya que existen diversas tendencias dentro de este movimiento, las cuales le aportan dinamismo, riqueza, pluralidad y complejidad, generando a menudo interesantes debates que abarcan cuestiones legales, técnicas, económicas o ideológicas. La *Open Source Initiative* (OSI, Iniciativa por el Código Abierto), por ejemplo, surgida a finales de la década de 1990, es partidaria de usar el término *open source* o "de código abierto" en lugar de "software libre", para el cual da una definición diferente que pretende tener más en cuenta los aspectos técnicos y no tanto las cuestiones morales o éticas como la FSF. En cualquier caso, aunque las filosofías y motivaciones de ambas organizaciones son diferentes, ambas comparten en la práctica la mayoría de las licencias que afectan a este tipo de productos. En un intento por unir los mencionados conceptos, dado que a menudo se refieren a productos semejantes, se está extendiendo el uso de la palabra "FLOSS" (*free/libre and open source software*, software libre y de código abierto) como término imparcial que engloba a los anteriores.

Por otra parte, es conveniente destacar que en ningún caso las definiciones de software libre o de código abierto implican que dichos productos sean gratuitos, si bien es frecuente que muchas de estas herramientas software se distribuyan de manera gratuita o a un coste notablemente inferior que las herramientas de software privativas (aunque al mismo tiempo también existen herramientas de software privativo gratuito).

- **Licencias de software libre**

Las licencias son autorizaciones formales con carácter contractual que los autores del software dan a un interesado para ejercer actos de explotación legales. Las licencias "libres" surgieron como una evolución de los derechos de autor o *copyright* (aplicados desde hace siglos a obras literarias, artísticas y científicas), para permitir dotar de respaldo legal a los programas desarrollados con la filosofía del software libre (en ocasiones se utiliza el término *copyleft* para designar genéricamente a este tipo de licencias).

En función de las diferentes condiciones bajo las cuales los autores gestionan los derechos de explotación de su obra y permiten a los usuarios redistribuir un programa u obras derivadas de éste, existen numerosos tipos de licencias libres y cada una tiene sus implicaciones técnicas, económicas y legales. Las más utilizadas son las aprobadas por la FSF, la OSI y algunas otras instituciones, como la Mozilla Foundation o el LaTex Project, las cuales pueden ser compatibles o no entre ellas. La primera licencia de sofware libre que apareció fue la *GNU General Public License*[4] (GPL, Licencia Pública General de GNU) y ha constituido un estándar de facto en este campo, si bien es de las más "restrictivas", pues obliga a liberar en las mismas condiciones toda obra derivada siquiera parcialmente de ella. Algunas otras licencias habitualmente utilizadas son la LGPL, la BSD, la MPL o la Apache License.

4 El software desarrollado en el presente trabajo ha sido licenciado bajo GPL v3. Puede verse el texto completo de la licencia en el Anexo de esta memoria

- **¿Por qué software libre?**

1) Por una parte, el software libre presenta una serie de **ventajas prácticas** desde el punto de vista técnico y económico:

   - Por lo general tiene un costo de adquisición más bajo que el software equivalente propietario, pues las empresas que desarrollan software libre con fines comerciales basan su negocio en mayor medida en ofrecer servicios alrededor de ese software (instalación, mantenimiento, soporte, etc.) y no tanto en la venta del software en sí, como por ejemplo es el caso de los sistemas operativos libres utilizados en la gran mayoría de servidores a nivel mundial.

   - Al permitir a otros muchos programadores leer, modificar y redistribuir el código fuente de un programa, éste tiende a evolucionar, desarrollarse y mejorar más rápidamente. También la detección y corrección de errores se produce antes. Esto se traduce en productos más eficientes, estables y robustos.

   - Además el software libre tiende a ser más diverso, pues las personas que contribuyen a modificar el programa lo utilizarán para solucionar sus propias necesidades, lo cual hace que el software tienda a estar adaptado a una cantidad mayor de problemas.

   - Favorece la reutilización del conocimiento, pues quien quiera desarrollar un programa podrá aprovechar el trabajo que otros ya han sintetizado en otro software, y no tendrá que partir de cero.

   - El ataque de virus informáticos contra el software libre es poco habitual, pues en caso de producirse, la comunidad que respalda el software libre será capaz de neutralizarlo mucho más rápidamente, con lo cual normalmente no compensará el esfuerzo de programar el código malicioso.

   - Proporciona independencia al cliente con respecto al proveedor. En caso de que su proveedor de software decida no continuar dándole soporte, o bien, por la razón que fuera, las condiciones ya no sean de su agrado, el cliente podrá acudir a otros proveedores que serán capaces de continuar el proyecto, al disponer del código fuente.

   - Favorece el desarrollo de la industria local. Con el uso y desarrollo de software libre, los profesionales tienen más posibilidades de aprender y mejorar para poder aportar valor añadido a sus actividades de manera tecnológicamente independiente, fomentando así el desarrollo del tejido industrial a nivel local.

   - Las soluciones de software libre suelen ser más fácilmente adaptables a requisitos de hardware más restrictivos, por lo que son más adecuadas para implantarse en equipos antiguos o con pocos recursos.

2) Por otra parte, el uso y promoción del software libre posee un conjunto de **ventajas éticas**:

   - Promueve modelos de relaciones humanas basadas en valores como la cooperación o la solidaridad, y no sólo en el puro mercantilismo. Las transacciones comerciales dentro del software libre tienden a buscar mayor justicia y no sólo el puro beneficio económico.

   - Defiende el conocimiento como un patrimonio colectivo que no se puede privatizar, ya que por un lado todo el mundo debe tener acceso a él, y por otro lado apropiarse de él sería injusto, pues todo nuevo aporte al conocimiento se aprovecha de los aportes hechos anteriormente y por tanto está en deuda con ellos.

   - Coloca a las TICs en una posición de responsabilidad en relación al devenir de la Humanidad, y de toma de conciencia sobre las consecuencias futuras de las acciones del presente.

   - Defiende la libertad como un derecho fundamental de los seres humanos.

Ambos conjuntos de ventajas fueron consideradas de gran importancia para este trabajo. Por una parte, debido a su validez en términos generales; por otra, por el hecho de que el presente trabajo se llevó a cabo en Cuba, donde algunas de las bondades del software libre cobran un valor todavía mayor, como por ejemplo las dos últimas ventajas prácticas mencionadas. Por estas razones, se eligió todo el software involucrado en la realización de este proyecto con el criterio de que fuera libre y de código abierto.

3.3 Compilador cruzado ARM GCC

- **Host y target**

Como se explicó anteriormente, el objetivo último de este proyecto es desarrollar una aplicación para el dispositivo Mini2440, con arquitectura ARM. Este dispositivo, que albergará y ejecutará la aplicación, es conocido como “blanco” o *target* de la aplicación.

La manera de crear dicha aplicación es utilizando una computadora auxiliar, conocida como anfitrión o *host*, en la que se escribe el código fuente del programa y se hace la compilación. Normalmente el host es una computadora de propósito general, por lo que en general no tendrá la misma arquitectura que el target. El código fuente debe ser compilado en la arquitectura host, pero de acuerdo a las reglas de compilación de la arquitectura target, para crear un fichero binario ejecutable por esta última. El programa que hace este tipo de compilación se conoce como *cross-compiler* o compilador cruzado.

En este proyecto se utilizó como host una PC portátil ASUS de la serie UL30A, con un microprocesador Intel Core 2 Duo, con arquitectura x86 de 64 bits, como se muestra en la tabla 3.2.

	Máquina	Microprocesador	Arquitectura	Tamaño registro
Target	Placa Mini2440 de FriendlyARM	Samsung S3C2440A	ARM920T (ARM)	32 bits
Host	PC portátil ASUS serie UL30A	Intel Core 2 Duo	x86-64 (x86)	64 bits

Tabla 3.2 *Dispositivos host y target en este proyecto*

- **ARM GCC**

GCC (*GNU Compiler Collection*, colección de compiladores GNU) es un conjunto de compiladores desarrollados por el proyecto GNU. Estos compiladores se consideran estándar para los sistemas operativos derivados de UNIX, de código abierto y algunos sistemas propietarios, como Mac OS X. GCC requiere el conjunto de aplicaciones conocido como *binutils* para realizar tareas como identificar archivos objeto u obtener su tamaño para copiarlos, traducirlos, enlazarlos o quitarles símbolos innecesarios. En su versión 4.3 incluye compiladores para C (gcc), C++ (g++), Java (gcj), Ada (GNAT), Objective-C (gobjc), Objective-C++ (gobjc++) y Fortran (gfortran). También está disponible, aunque no de forma estándar, soporte para Go (gccgo), Modula-2, Modula-3, Pascal (gpc), PL/I, D (gdc), Mercury y VHDL (ghdl).

ARM GCC es la extensión de GCC para la compilación cruzada para arquitecturas ARM. Tanto GCC como ARM GCC son software libre y son distribuidos por la FSF bajo la licencia GPL.

En este proyecto se programó en lenguaje C++, por tanto la herramienta GCC utilizada fue g++.

3.4 Kernel de Linux

El kernel de Linux es el núcleo de todos los sistemas operativos de la familia GNU/Linux, los cuales constituyen una subfamilia dentro de la familia de sistemas operativos *UNIX-like* (similares a UNIX). El núcleo de un sistema operativo es su parte más importante, ya que es responsable de la gestión de los recursos hardware de la computadora (memoria, procesador, periféricos...), así como de la comunicación entre los distintos programas con el hardware y entre sí.

La primera versión del kernel de Linux fue desarrollada en 1991 para una computadora i386 por el entonces estudiante finlandés Linus Torvalds, quien distribuyó el programa bajo licencia GPL v2. Este hecho permitió, gracias al auge de internet, que en poco tiempo se extendiera a una enorme cantidad de usuarios y programadores que tuvieron la

oportunidad de usarlo, acceder a su código fuente y aportar cambios y mejoras. La unión de aquel kernel con otras herramientas de sistema desarrolladas por el proyecto GNU dio lugar a los sistemas operativos libres conocidos como GNU/Linux, de los que existen infinidad de distribuciones y que hoy en día constituyen un estándar. Gracias al aporte de desarrolladores de todo el mundo, actualmente el kernel de Linux proporciona soporte para la mayoría de arquitecturas hardware existentes, una de las cuales es ARM.

Las características del kernel de Linux hacen que sea particularmente adecuado para los sistemas embebidos, donde es ampliamente utilizado debido a su reducido tamaño –una instalación típica de Linux embebido ocupa alrededor de 2MB, frente a los 21MB de Windows CE–, adaptabilidad, estabilidad o respaldo, entre otras razones. En el campo de los sistemas operativos móviles, por ejemplo, los sistemas basados en Linux (Android, Ångström, Maemo...) representan actualmente más de una tercera parte de la cuota de mercado mundial.

En este trabajo se utilizó un kernel de Linux como núcleo del sistema operativo de la placa Mini2440 y de la PC host. Las herramientas del kernel empleadas fueron principalmente comandos de la *shell* bash y el sistema de archivos. En menor medida también se tuvo que tratar con los sistemas de gestión de procesos y de memoria y el *framebuffer*.

- **Sistema de archivos**

El sistema de archivos de Linux tiene la siguiente estructura básica:

```
/                       Directorio raíz
    |-- bin/            Binarios de comandos esenciales
    |-- boot/           Archivos estáticos de boot loader
    |-- dev/            Archivos de dispositivos
    |-- etc/            Configuración del sistema local-máquina
    |-- home/           Directorios hogar de los usuarios
    |-- lib/            Librerías compartidas
    |-- mnt/            Punto de montaje de particiones temporales
    |-- root/           Directorio hogar del usuario root
    |-- sbin/           Binarios del sistema esenciales
    |-- tmp/            Archivos temporales
    |-- usr/            Jerarquía secundaria de datos de usuario
    |     |-bin/        Comandos binarios no para todos los usuarios
    |     |-include/    Archivos de cabecera
    |     |-lib/        Bibliotecas compartidas de binarios en /usr/bin/
    |     |-sbin/       Sistema de binarios no esenciales
    |     |-share/      Arquitectura independiente y compartida de datos
    |     |-src/        Códigos fuente de algunas aplicaciones
    |     |-X11R6/      Reservado para el entorno gráfico X Window
    |     |-local/      Jerarquía terciaria para datos locales
    |
    |-- var/            Información variable
```

- **Comandos bash**

`ls`	Muestra un listado del contenido de un directorio
`cd`	Cambia de directorio
`mkdir`	Crea un nuevo directorio
`su`	Entra en modo super-usuario
`sudo`	Ejecuta con privilegios de root
`ifconfig`	Configura la interfaz de red
`tar`	Comprime y descomprime paquetes
`chmod`	Cambia los permisos de un archivo
`chown`	Cambia el propietario de un archivo
`telnet`	Inicia una conexión telnet
`ftp`	Inicia una conexión FTP

3.5 Qt y Qtopia

- **Qt**

Qt es una biblioteca multiplataforma diseñada principalmente para el desarrollo de aplicaciones con interfaces gráficas de usuario. Ha sido utilizada para escribir programas como Skype, Google Earth, VLC media player o el entorno de escritorio KDE. Utiliza el lenguaje de programación C++ y funciona en todas las principales plataformas (GNU/Linux, Windows, Mac OS...).

Su primera versión fue desarrollada en 1992 por la empresa noruega Trolltech. En 2008 fue adquirida por la empresa finlandesa Nokia, que durante unos años hizo una fuerte inversión en este producto. Sin embargo, a comienzos de 2011, el negocio de licenciamiento comercial fue vendido a otra empresa finlandesa, Digia, aunque Nokia tiene intención de seguir respaldando el proyecto.

Actualmente se distribuye en tres ediciones con diferentes licencias: GPL, LGPL y QPL (esta última propietaria), puediendo ser gratuitas o de pago y para fines comerciales o no comerciales.

Además de su amplio repertorio de métodos y funciones, la biblioteca Qt incluye una serie de herramientas muy útiles para el desarrollo de aplicaciones. Las que se utilizaron en este proyecto son las siguientes:

- qmake: herramienta que automatiza la creación de ficheros Makefile

- Meta Object Compiler (moc): herramienta para generar ficheros de código fuente para el compilador de C++ a partir de los ficheros fuente que definen procesos de eventos (SIGNAL/SLOT)

- Qt Virtual Framebuffer (QVFb): herramienta que permite simular el comportamiento de la aplicación a través de un “target virtual” en el host. Posee diferentes apariencias personalizables en función del tipo de target al que vaya destinada la aplicación (teléfonos móviles, PDAs, etc.)

Figura 3.1 *QVFb ejecutándose en el host con apariencia de Greenphone*

- **Qtopia**

Qtopia (conocido como Qt Extended a partir de octubre de 2008) es una plataforma de aplicaciones para dispositivos móviles basados en Linux. Qtopia forma una capa de abstracción que trabaja por encima del kernel de Linux proporcionando servicios como un entorno gráfico basado en ventanas, pantalla de escritura manual, aplicaciones de información personal, multimedia, videojuegos, internet, sincronización con PC... El conjunto “kernel de Linux + Qtopia” se puede ver como un sistema operativo embebido con entorno gráfico.

Fue creado por la empresa Qt Software (antes Trolltech), que desarrollaba dos categorías de Qt Extended, una libre, bajo licencia GPL, y otra propietaria; así como dos ediciones, una para teléfonos móviles y otra para PDAs. Qt Software canceló el desarrollo de Qt Extended, pero, como era software libre, a partir de él la comunidad creó el proyecto Qt Extended Improved y continuó construyendo.

En este proyecto se trató principalmente con su sistema de archivos, para alojar adecuadamente la aplicación final e integrarla así en la plataforma de aplicaciones que ofrece el sistema operativo. La estructura del sistema de archivos de Qtopia, que queda encajada en el sistema de archivos de Linux, es la siguiente:

```
/opt/Qtopia/
        |-- apps/
        |      |-Applications/
        |      |-FriendlyARM/
        |      |-Games/
        |      |-Settings/
        |
        |-- pics/
        |-- bin/
        |-- help/
```

4. DESARROLLO DE LA APLICACIÓN

4.1 Introducción

En este capítulo se describe paso por paso el procedimiento seguido hasta conseguir la ejecución de la aplicación final en la placa Mini2440: en primer lugar se explicará cómo se estableció el entorno de desarrollo, utilizando las herramientas descritas en los capítulos anteriores; a continuación se describirá detalladamente el proceso de desarrollo en sí y por último se presentará el código fuente programado**.**

4.2 Construcción del entorno de desarrollo

Para poder crear la aplicación, previamente es necesario construir un entorno de desarrollo adecuado. Para ello se deberán hacer ciertas configuraciones en los dos dispositivos involucrados (host y target), así como establecer una comunicación entre ellos mediante protocolos convenientes, que permita la transferencia de archivos del uno a otro. A continuación se describe el proceso seguido para la construcción de dicho entorno de desarrollo, el cual queda resumido en la figura 4.1.

4.2.1 Configuración del target

En este trabajo se empleó la configuración por defecto de la placa Mini2440, que consiste en una instalación del bootloader Supervivi en la memoria NOR Flash, y una instalación de la versión 2.6.29 del kernel de Linux, configurado con los drivers correspondientes para los dispositivos de la placa, junto con la plataforma Qtopia 2.2.0, en la NAND Flash. Por lo tanto, no hubo que hacer ninguna configuración adicional en este dispositivo.

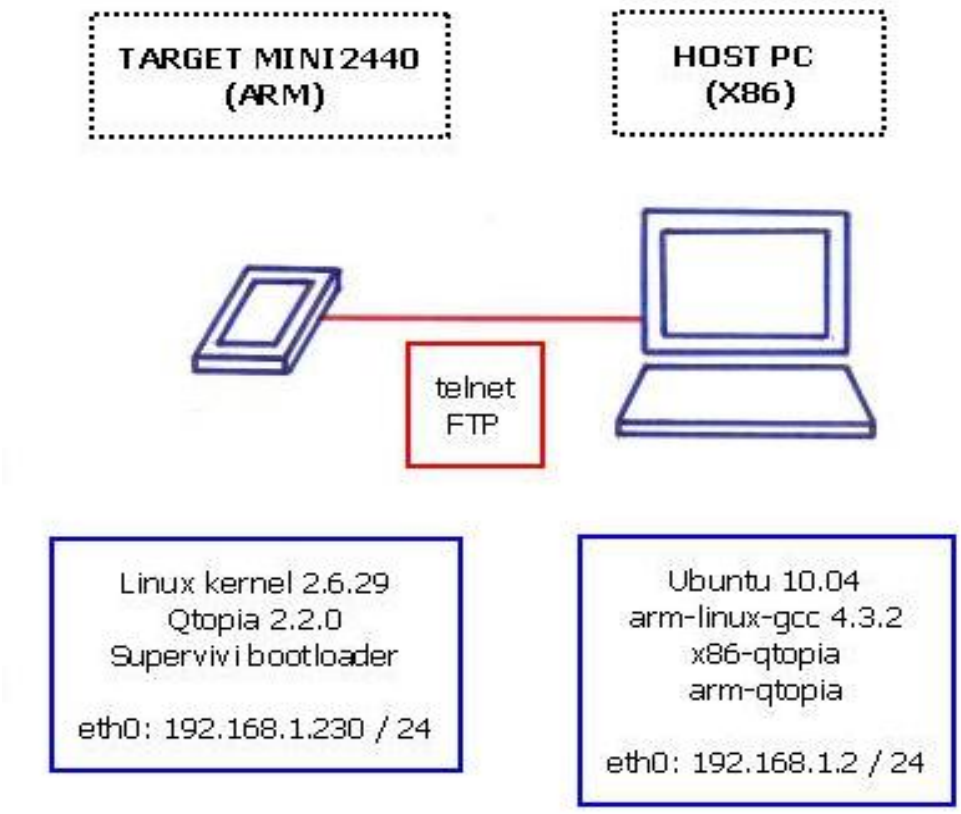

Figura 4.1 *Entorno de desarrollo*

4.2.2 Configuración del host

- **Sistema operativo**

En primer lugar se debe instalar en la PC host un sistema operativo GNU/Linux. En este trabajo se utilizó la distribución Ubuntu 10.04 LTS, elegida por su facilidad de manejo, gratuidad, estabilidad, comodidad de instalación (desde CD autoarrancable) y amplio soporte en relación al número de usuarios en la comunidad y al software disponible. En cualquier caso, todo el proceso que sigue sería prácticamente idéntico en el caso de elegir otras distribuciones populares como Debian, Fedora u openSUSE.

Una vez instalado y actualizado el sistema operativo, se deberá instalar una serie de paquetes y dependencias adicionales de desarrollo. Para ello se abre un terminal y se ejecuta desde la línea de comandos:

```
$ sudo apt-get install g++ libx11-dev libxmu-dev libxext-dev
    zlib1g-dev libjpeg62-dev libssl-dev uuid-dev
```

Por otra parte, se necesitará disponer de un editor de texto para escribir el código fuente del programa. Si bien la instalación de Ubuntu 10.04 incluye varios editores de texto, se consideró oportuno utilizar el editor geany, que tiene algunas funcionalidades de entorno de desarrollo integrado (IDE) para varios lenguajes de programación, incluido C++.

```
$ sudo apt-get install geany
```

- **Compilador cruzado**

El siguiente paso es instalar el compilador cruzado. En este caso se utilizó la versión 4.3.2 del compilador GCC para ARM, que está incluido en el DVD del kit, en el directorio /linux, en el fichero comprimido arm-linux-gcc-4.3.2 .tgz. Para instalarlo primero se deberá copiar el fichero (como super usuario) a un directorio conveniente, como por ejemplo /tmp. A continuación se accede a esa carpeta y se extrae el contenido del fichero:

```
$ cd /tmp
$ chmod +x *.tgz
$ sudo tar xfzv arm-linux-gcc-4.3.2.tgz -C/
```

De esta manera el compilador cruzado queda instalado en el directorio /user/local/arm/4.3.2 . Por comodidad conviene añadir a la variable de entorno PATH la ruta de dicho directorio. Para ello se edita el contenido del archivo bash.bashrc con un editor de texto, por ejemplo gedit

```
$ sudo gedit /etc/bash.bashrc
```

añadiendo la siguiente línea:

```
PATH=$PATH:/usr/local/arm/4.3.2/bin
```

Para comprobar que el compilador está correctamente instalado, se deberá reiniciar la sesión de usuario, y a continuación:

```
$ arm-linux-gcc -v
```

Si la instalación fue correcta, se debe obtener por pantalla:

```
[...]
Thread model: posix
gcc version 4.3.2 (Sourcery G++ Lite 2008q3-72)
```

- **x86-qtopia y arm-qtopia**

Además del compilador cruzado, es necesario instalar los paquetes x86-qtopia y arm-qtopia, los cuales proporcionan soporte para el desarrollo de aplicaciones en Qt y para su ejecución en las arquitecturas x86 y ARM, respectivamente. Los ficheros comprimidos se encuentran en el directorio /linux del DVD, con los nombres de x86-qtopia.tgz y arm-qtopia.tgz.

Primero se deben copiar ambos ficheros a un directorio conveniente, por ejemplo /tmp. Desde ese directorio, se extrae en primer lugar el contenido del fichero x86-qtopia al directorio /opt/FriendlyARM/mini2440:

```
$ cd /tmp
$ sudo tar xfvz  x86-qtopia.tar.gz -C /opt/FriendlyARM/mini2440
```

A continuación se extraen los archivos comprimidos individuales:

```
$ cd /opt/FriendlyARM/mini2440/x86-qtopia/
$ sudo chmod +x *.gz
$ sudo tar xfz qtopia-2.2.0.tar.gz
$ sudo tar xfz konq.tar.gz
$ mkdir qtopia-2.2.0-FriendlyARM/qtopia/image
$ sudo tar xfvz fonts.tar.gz -C qtopia-2.2.0-
     FriendlyARM/qtopia/image
```

Para poder realizar la instalación correctamente, el usuario debe pasar a ser el propietario de todo el contenido del directorio:

```
$ sudo chown -R [usuario]:[grupo_del_usuario] /opt/FriendlyARM/
```

Por último, se ejecuta el script build-all, que instalará los paquetes:

```
$ ./build-all
```

Se sigue un procedimiento similar para arm-qtopia:

```
$ cd /tmp
$ sudo tar xfz arm-qtopia.tgz -C /opt/FriendlyARM/mini2440/
$ cd /opt/FriendlyARM/mini2440/
$ sudo chmod +x *.gz
$ sudo tar xfz qtopia-2.2.0.tar.gz
$ sudo tar xfz konq.tar.gz
$ sudo tar xfz fonts.tar.gz
$ sudo chown -R [usuario]:[grupo_del_usuario] /opt/FriendlyARM/
$ ./build-all
```

4.2.3 Comunicación entre los dispositivos

Para permitir la comunicación entre el host y el target es necesario configurar una sencilla red LAN. La dirección IP por defecto de la placa Mini2440 es 192.168.1.230, con máscara de red 255.255.255.0. Configurando la tarjeta de red de la PC con la dirección IP 192.168.1.2 y la misma máscara, y utilizando un cable Ethernet cruzado para conectar ambos dispositivos, ya es posible la comunicación entre ellos por medio de los protocolos FTP y telnet (puertos 21 y 23 respectivamente).

```
$ sudo ifconfig eth0 192.168.1.2 netmask 255.255.255.0
```

FTP se utilizará para transferir ficheros desde el host a la placa, como se verá más adelante. Telnet es conveniente para ejecutar terminales del Mini2440 cómodamente desde el host, como por ejemplo para establecer la contraseña del root.

4.3 Proceso del desarrollo

Una vez construido el entorno de desarrollo, se está en condiciones de llevar a cabo el desarrollo de la aplicación propiamente dicho. La aplicación, que constituye el objetivo final de este proyecto, será en última instancia un fichero binario ejecutable por la arquitectura ARM que, alojado adecuadamente en el sistema de archivos de Qtopia (junto con algún otro fichero de control y una imagen que servirá de icono de lanzamiento de la aplicación), quedará integrado en el conjunto de herramientas que ofrece el sistema operativo del Mini2440.

Para el desarrollo de la aplicación se siguió el procedimiento por pasos que se explica a continuación y que queda resumido en el diagrama de flujo de la figura 4.2:

1) Utilizando un editor de texto, en este caso geany, se escribe el código del programa en lenguaje C++, generando así los ficheros (.cpp y .h) que constituyen el **código fuente** de la aplicación.

2) Utilizando el mismo editor de texto se crea el **fichero de proyecto** (.pro), definiendo los parámetros TEMPLATE, HEADERS, SOURCES, etc. El fichero de proyecto final de este trabajo (grafica.pro) contiene lo siguiente:

```
TEMPLATE  =  app
CONFIG    =  qt warn_on release
HEADERS   =  window.h \
             widget.h
SOURCES   =  main.cpp \
             widget.cpp \
             window.cpp
TARGET    =  grafica
```

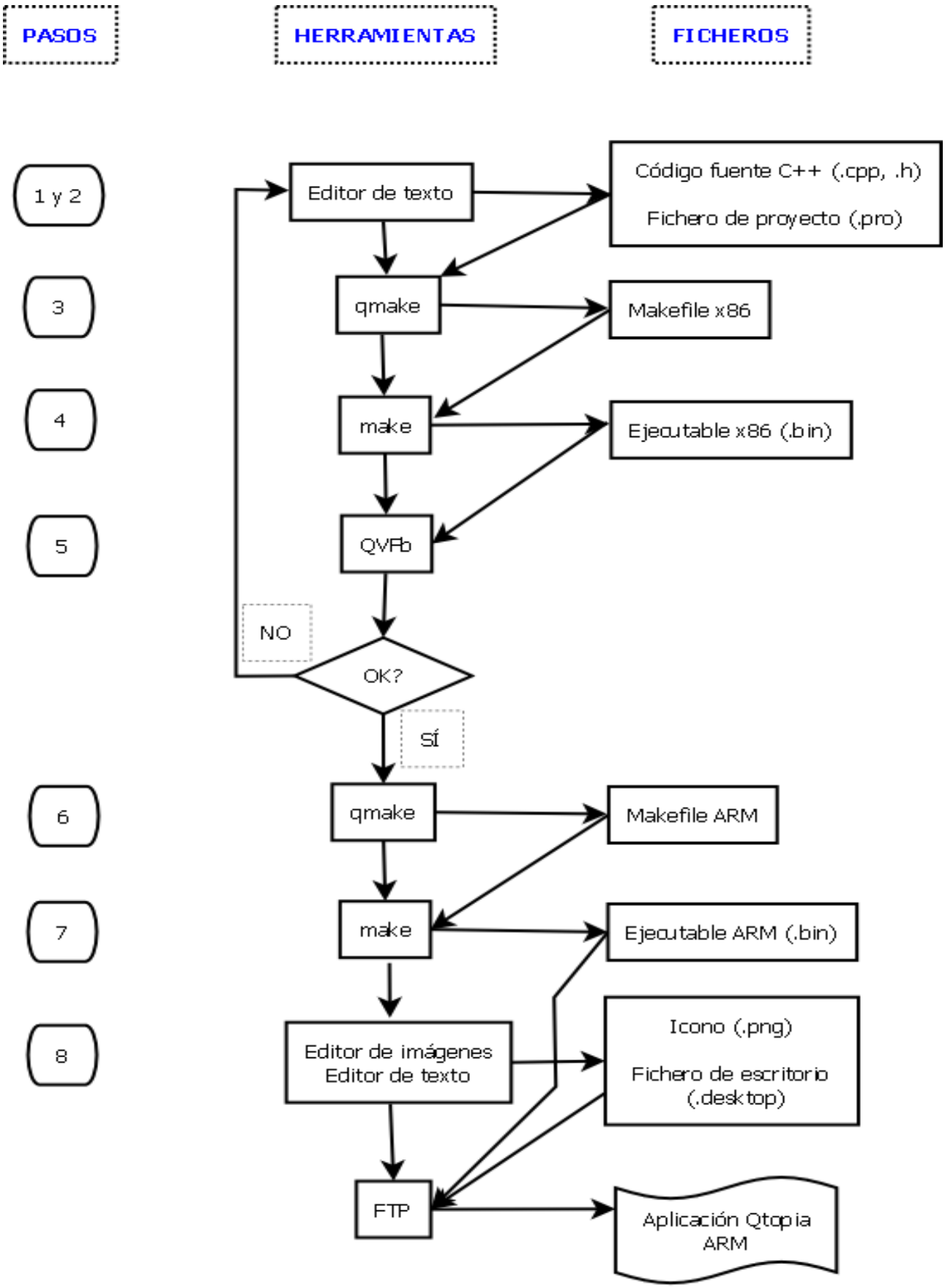

Figura 4.2 *Proceso de desarrollo*

3) Se ejecuta en el host el binario anterior y mediante la herramienta QVFb se simula el comportamiento de la aplicación en la arquitectura ARM. Si el funcionamiento de la aplicación es correcto, se salta al paso siguiente. Si no, se deberá regresar al primer paso y depurar el código fuente de la aplicación.

4) Mediante la herramienta qmake se genera el **fichero makefile** para el **ARM**:

```
$ source /opt/FriendlyARM/mini2440/arm-qtopia/qtopia-2.2.0-
    FriendlyARM/setQpeEnv

$ qmake -spec /opt/FriendlyARM/mini2440/arm-qtopia/qtopia-
    2.2.0-FriendlyARM/qtopia/mkspecs/qws/linux-arm-g++ -o
    Makefile *.pro
```

5) Mediante la herramienta make se crea el **fichero binario ejecutable** (.bin) para el **ARM**:

```
$ make clean
$ make
```

6) Mediante una conexión FTP entre los dispositivos a través de un cable Ethernet cruzado, se descargará la aplicación del host al target. Para ello será necesario copiar tres archivos diferentes y ubicarlos en respectivos directorios del sistema de archivos del Qtopia instalado en el Mini2440.

- El binario ejecutable ARM .bin debe ser copiado al directorio /opt/Qtopia/bin/, que contiene los ejecutables de todas las aplicaciones de Qtopia. En este caso, el fichero se llamará grafica.bin.

- Un **fichero de icono** (.png), que servirá para lanzar la aplicación al hacer click sobre él, debe ser copiado al directorio /opt/Qtopia/pics/. El icono puede crearse mediante un editor de imágenes o utilizando el método que se considere oportuno. Debe tener el mismo nombre que la aplicación, es decir, en este caso, grafica.png.

Figura 4.3 *Icono "grafica.png"*

- Un **fichero de escritorio** (.desktop), que contiene información para el lanzador de aplicaciones de Qtopia, tal como el nombre del ejecutable, el tipo de paquete, el icono, el nombre de la aplicación que saldrá en el menú de pantalla, etc., debe ser copiado a un directorio con el mismo nombre creado dentro del directorio /opt/Qtopia/apps/Applications/. Este fichero es creado mediante un editor de texto y debe tener el mismo nombre que los anteriores. En este caso se llamará grafica.desktop y su contenido es el siguiente:

```
[desktop entry]
Name=Gráfica
Exec=grafica
Type=Application
Icon=grafica
Comment=Representa señales gráficamente
```

La descarga por FTP de los tres ficheros (grafica.bin, grafica.png y grafica.desktop) del host al target se realiza de la siguiente manera:

```
$ su
# ftp 192.168.1.230
   Name: root
   Password: XXXX
ftp> cd opt/Qtopia/bin
ftp> send grafica.bin
ftp> cd /opt/Qtopia/pics
ftp> send grafica.png
ftp> cd /opt/Qtopia/apps
ftp> mkdir grafica
ftp> cd grafica
ftp> send grafica.desktop
ftp> bye
```

Finalmente, los ficheros quedan alojados así en el sistema de archivos:

```
/opt/Qtopia/
        |-- apps/
        |     |-Applications/  → grafica.desktop (fichero de información)
        |     |-FriendlyARM/
        |     |-Games/
        |     |-Settings/
        |
        |-- pics/              → grafica.png (icono)
        |-- bin/               → grafica.bin (binario ejecutable)
        |-- help/
```

Al reiniciar el Mini2440, en la pestaña “Applications” se habrá añadido el icono de nuestra aplicación, con el nombre “Gráfica”. Para lanzar la aplicación se hace click sobre el icono y ésta comenzará a ejecutarse. Una vez lanzada, se pueden utilizar los tres botones de la esquina superior derecha de la barra de título de la aplicación, para minimizarla, maximizarla/restaurarla y cerrarla, respectivamente.

4.4 Código fuente del programa

El código fuente de la aplicación final está compuesto por cinco ficheros: el fichero principal (main.cpp) y dos auxiliares (widget.cpp y window.cpp), junto con sus respectivos ficheros de cabecera (widget.h y window.h). En los ficheros de cabecera se declaran las clases utilizadas, sus herencias, constructores, destructores, métodos y variables, públicos, privados o protegidos, los cuales se definen en los ficheros .cpp.

El código del programa es el siguiente:

- Fichero main.cpp

```
/*************************************************************************

(C) 2011 Fernando Marcotegui ////////////////////////////////

Programa para representar gráficamente señales eléctricas en el Mini2440

Este programa es software libre y se distribuye bajo los términos de la
Licencia Pública General de GNU (GPL) en su versión 3. Más información sobre
esta licencia en: http://www.gnu.org/licenses/gpl.html

*************************************************************************/

#include <qapplication.h>
#include "window.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );
    MainWindow *win = new MainWindow();

    win->showMaximized();

    a.setMainWidget( win );

    win->main_box->setMinimumHeight(win->height());
    win->main_box->setMaximumHeight(win->height());
    win->widget->setMinimumWidth(win->width()-win->control_box2->width());
    win->widget->setMaximumWidth(win->width()-win->control_box2->width());

    win->show();

    a.exec();

    return 0;
}
```

- Fichero widget.cpp

```
/*************************************************************************

(C) 2011 Fernando Marcotegui ////////////////////////////////

Programa para representar gráficamente señales eléctricas en el Mini2440
```

```
Este programa es software libre y se distribuye bajo los términos de la
Licencia Pública General de GNU (GPL) en su versión 3. Más información sobre
esta licencia en: http://www.gnu.org/licenses/gpl.html

*****************************************************************************/

#include "widget.h"
#include <qstring.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/fs.h>
#include <errno.h>
#include <string.h>
#include <math.h>

ConnectWidget::ConnectWidget( QWidget *parent, const char *name )
    : QWidget( parent, name, 0 )
{
      setBackgroundColor( white );
      max = 0.0;
      time_fresh = 1;
      myPenWidth = 1;
      last_x = 0;
      last_y = 0;
      zoom = 1;
      pause = false;
      paint = new QPainter ( this );
      device = open("/dev/adc", 0);
      if (device < 0)
            perror("open ADC device:");
}

ConnectWidget::~ConnectWidget()
{
      delete[] paint;
      close(device);
}

void ConnectWidget::paintEvent( QPaintEvent * )
{
      if ( !pause )
            addToBuffer();
      drawLinesInit ( paint );
      sleep (time_fresh);
      paint->setPen( QPen(Qt::red, 1, Qt::SolidLine) );
      for ( int i = MAX_VALUE-1, x = width()-1 ; x >= 0 ; x--, i-- )
            paint->drawLine   (x,   height()-(buffer[i]*zoom),   x-1,   height()-
(buffer[i-1]*zoom));

}

void ConnectWidget::addToBuffer( )
{
      for ( int i = 0; i < MAX_VALUE-1; i++ )
            buffer[i] = buffer[i+1];
      int value;
      char buff[30];
      int len = read(device, buff, sizeof buff -1);
      if (len > 0)
```

```
	{
		buff[len] = '\0';
		value = -1;
		sscanf(buff, "%d", &value);
	}
	else
		perror("read ADC device:");
	buffer[MAX_VALUE-1] = value/4;
}

void ConnectWidget::drawLinesInit( QPainter* paint )
{
	int x = width(), y = height();
	paint->setPen(QPen(Qt::green,  myPenWidth,  Qt::DashLine,  Qt::RoundCap,
Qt::RoundJoin));

	paint->drawLine(0, y/2, x, y/2);
	paint->drawLine(0, y/4, x, y/4);
	paint->drawLine(0, 3*y/4, x, 3*y/4);
	paint->drawLine(x/2, 0, x/2, y);
	paint->drawLine(x/4, 0, x/4, y);
	paint->drawLine(3*x/4, 0, 3*x/4, y);
	paint->setPen(Qt::black);
	paint->setFont(QFont("Arial", 13));
	QString str;

	paint->drawText(x/4+5, y/4-5, str.setNum(3*max/(4*zoom)));
	paint->drawText(x/4+5, y/2-5, str.setNum(max/(2*zoom)));
	paint->drawText(x/4+5, 3*y/4-5, str.setNum(max/(4*zoom)));
	paint->drawText(x/2+5, y/4-5, str.setNum(3*max/(4*zoom)));
	paint->drawText(x/2+5, y/2-5, str.setNum(max/(2)));
	paint->drawText(x/2+5, 3*y/4-5, str.setNum(max/(4)));
	paint->drawText(3*x/4+5, y/4-5, str.setNum(3*max/(4*zoom)));
	paint->drawText(3*x/4+5, y/2-5, str.setNum(max/(2*zoom)));
	paint->drawText(3*x/4+5, 3*y/4-5, str.setNum(max/(4*zoom)));
	update();
}
```

- Fichero window.cpp

```
/****************************************************************************

(C) 2011 Fernando Marcotegui ////////////////////////////////

Programa para representar gráficamente señales eléctricas en el Mini2440

Este programa es software libre y se distribuye bajo los términos de la
Licencia Pública General de GNU (GPL) en su versión 3. Más información sobre
esta licencia en: http://www.gnu.org/licenses/gpl.html

****************************************************************************/

#include "window.h"


MainWindow::MainWindow( QWidget *parent, const char *name )
    : QWidget( parent, name, 0 )
{
	main_box = new QHBox ( this );

	control_box = new QVBox ( main_box );
	control_box->setSpacing(6);
```

```
	control_box->setMargin(6);
	control_box = new QVBox ( main_box );

	boton = new QPushButton(control_box);
	boton->setText ( "Pausar" );
	boton->setToggleButton ( true );

	control_box2 = new QHBox ( control_box );
	x_box = new QVBox ( control_box2 );
	x_box->setFrameStyle( QFrame::WinPanel | QFrame::Raised );
	text1 = new QLabel( x_box );
	text1->setTextFormat ( RichText );
	text1->setText ( "<b>Tiempo<b>" );
	x_min = new QLabel(x_box);
	x_min->setAlignment (AlignHCenter | AlignBottom );
	x_min->setTextFormat ( RichText );
	x_min->setText ( "0 sec" );
	x_ = new QSlider ( 0, 3, 1, 1, Qt::Vertical, x_box );
	x_->setTickmarks ( QSlider::Both );
	x_max = new QLabel(x_box);
	x_max->setAlignment ( AlignHCenter | AlignBottom );
	x_max->setTextFormat ( RichText );
	x_max->setText ( "3 sec" );

	y_box = new QVBox ( control_box2 );
	y_box->setFrameStyle( QFrame::WinPanel | QFrame::Raised );
	text2 = new QLabel(y_box);
	text2->setAlignment (AlignHCenter | AlignBottom );
	text2->setTextFormat ( RichText );
	text2->setText ( "<b>Zoom<b>" );
	y_min = new QLabel( y_box );
	y_min->setAlignment ( AlignHCenter | AlignBottom );
	y_min->setTextFormat ( RichText );
	y_min->setText ( "1x" );
	y_ = new QSlider ( 1, 5, 1, 1, Qt::Vertical, y_box );
	y_->setTickmarks ( QSlider::Both );
	y_max = new QLabel( y_box );
	y_max->setAlignment ( AlignHCenter | AlignBottom );
	y_max->setTextFormat ( RichText );
	y_max->setText ( "5x" );

	control_box->adjustSize();
	control_box2->adjustSize();

	widget = new ConnectWidget ( main_box );
	widget->max = 1024;
	main_box->adjustSize();

	connect( x_, SIGNAL( valueChanged( int ) ), this, SLOT( connectX () ) );
	connect( y_, SIGNAL( valueChanged( int ) ), this, SLOT( connectY () ) );
	connect( boton, SIGNAL( clicked() ), this, SLOT( setPause () ) );
}

void MainWindow::connectX ()
{
	widget->time_fresh = x_->value();
}

void MainWindow::connectY ()
{
	widget->zoom = y_->value();
}
```

```
void MainWindow::setPause ()
{
      widget->pause = !widget->pause;
      if ( widget->pause )
            boton->setText ( "Continuar" );
      else
            boton->setText ( "Pausar" );
}

MainWindow::~MainWindow ()
{
      delete[] main_box;
      delete[] control_box;
      delete[] control_box2;
      delete[] x_;
      delete[] y_;
      delete[] widget;
      delete[] x_box;
      delete[] y_box;
      delete[] main_text;
      delete[] boton;
}
```

- Fichero widget.h

```
/****************************************************************************

(C) 2011 Fernando Marcotegui ////////////////////////////////

Programa para representar gráficamente señales eléctricas en el Mini2440

Este programa es software libre y se distribuye bajo los términos de la
Licencia Pública General de GNU (GPL) en su versión 3. Más información sobre
esta licencia en: http://www.gnu.org/licenses/gpl.html

****************************************************************************/

#include <qwidget.h>
#include <qpainter.h>

const int MAX_VALUE = 10000;

class ConnectWidget : public QWidget
{
      Q_OBJECT

public:
      ConnectWidget( QWidget *parent=0, const char *name=0 );
      ~ConnectWidget();
      int time_fresh, zoom;
      double max;
      bool pause;

protected:
      void paintEvent( QPaintEvent * );

private:
      void drawLinesInit( QPainter * );
      void addToBuffer ();
      int myPenWidth, last_x, last_y, width_temp, height_temp,
buffer[MAX_VALUE];
    QPainter *paint;
```

```
    int device;

};
```

- Fichero window.h

```
/*************************************************************************

(C) 2011 Fernando Marcotegui////////////////////////////////

Programa para representar gráficamente señales eléctricas en el Mini2440

Este programa es software libre y se distribuye bajo los términos de la
Licencia Pública General de GNU (GPL) en su versión 3. Más información sobre
esta licencia en: http://www.gnu.org/licenses/gpl.html

*************************************************************************/

#include <qhbox.h>
#include <qvbox.h>
#include <qslider.h>
#include <qframe.h>
#include <qlabel.h>
#include <qpushbutton.h>
#include "widget.h"


class MainWindow : public QWidget
{
      Q_OBJECT

      public slots:
            void connectX ();
            void connectY ();
            void setPause ();

      public:
            MainWindow ( QWidget * parent = 0, const char * name = 0 );
            ~MainWindow ();

            ConnectWidget *widget;
            QSlider *x_, *y_;
            QPushButton *boton;

            QLabel *text1, *text2, *x_min, *x_max, *y_min, *y_max, *main_text;
            QHBox *main_box, *x_box, *y_box, *control_box2;
            QVBox *control_box;

};
```

RESULTADOS

Empleando el proceso descrito en el capítulo 4, se consiguió integrar una aplicación en la plataforma Qtopia de la placa Mini2440 tal que, al ser lanzada, maximiza una ventana en la que se dibujan unos ejes cartesianos, sobre los cuales se comienza a representar la evolución temporal de la tensión que existe en el pin AIN0 del conector CON4 de la placa, de derecha a izquierda de la pantalla. Como complemento, se logró añadir a la aplicación las funcionalidades más básicas de un osciloscopio, es decir, un control para regular la base de tiempos (“Tiempo”) y otro para el eje vertical (“Zoom”). Además, se le añadió un botón de captura de pantalla (“Pausar”) que detiene la imagen hasta volver a ser pulsado, para poder observar mejor la forma de onda en el momento deseado.

No obstante, mientras que los controles de captura y zoom funcionan correctamente, el control implementado para la base de tiempos no es del todo satisfactorio. Esto se debe a que la manera de modificar la base de tiempos es un tanto rudimentaria, ya que el diezmado se consigue “durmiendo” la rutina que pinta la gráfica en pantalla durante un tiempo mayor o menor. Este método funciona cualitativamente, sin embargo no se consiguió depurar el sistema lo suficiente como para dar resultados aceptables.

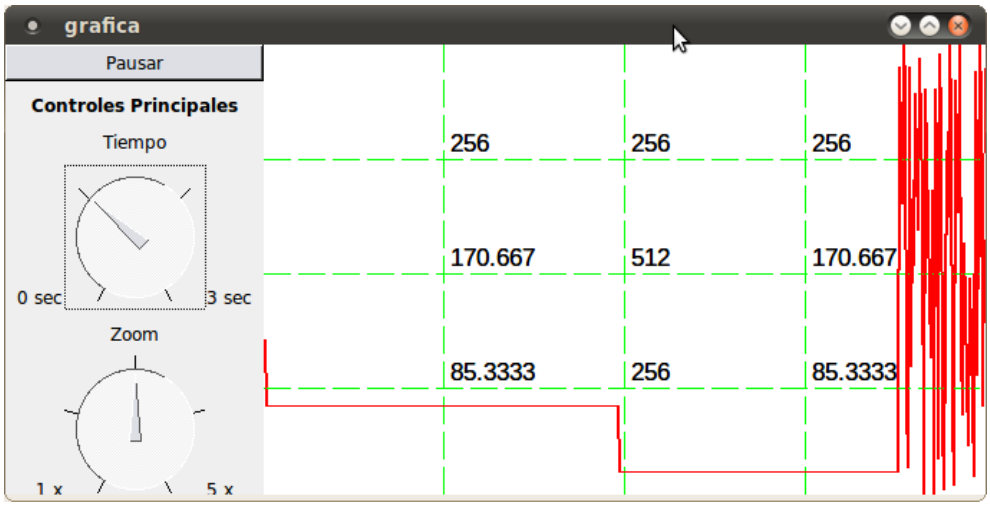

Figura 4.3 *Simulación de la aplicación en el host mediante QVFb*

Como recomendación para posibles trabajos futuros con esta herramienta, sería conveniente implementar un sistema de adquisición de datos que fuera más parecido al de un osciloscopio digital real, es decir, que funcione con un sistema de *trigger,* realice cierto procesado de la señal, como promediado móvil o detección de picos, y sobre todo que tenga mayor control sobre el diezmado para la base de tiempos. Si esto se consigue, se podría además intentar añadir un segundo canal a la gráfica (o incluso hasta tres canales más, manejando adecuadamente el puerto GPIO).

CONCLUSIONES

Observando los resultados obtenidos de este trabajo, se puede concluir lo siguiente:

- Se han cumplido el objetivo general y los objetivos específicos marcados al inicio del proyecto.

- Se ha demostrado que la placa Mini2440 es una herramienta adecuada para la representación gráfica de señales eléctricas en el dominio del tiempo.

- Se ha demostrado también que es posible realizar trabajos como éste de acuerdo con los principios del software libre y de código abierto y utilizando solamente programas de estas características.

- Se han sentado las bases para la utilización de este tipo de sistemas embebidos en el ámbito académico de la Universidad de Pinar del Río, Cuba.

BIBLIOGRAFÍA

[1] Barr, Michael. *Programming Embedded Systems in C and C++*. O'Reilly, 1999

[2] Daithankar, Shridhar. *A Tutorial for C/C++ Programming on Linux*. 2004

[3] Loosemore, Sandra *et ál*. *The GNU C Library Reference Manual.* FSF, 2007

[4] Schildt, Herbert. *C++. Guía de autoenseñanza*. McGraw-Hill, 1995

[5] Stallings, William. *Sistemas Operativos*. 4ª Ed. Pearson, 2003

[6] Bovet, Daniel P.; Cesati, Marco. *Understanding the Linux Kernel*. 3rd Ed. O'Reilly, 2005

[7] Culebro Juárez, Montserrat *et ál*. *Software libre vs software propietario. Ventajas y desventajas.* 2006

[8] *Mini2440 Hardware Essentials*. Industrial ARMWorks, 2009

[9] *Sharp SL-series Zaurus "Qtopia" Development Start-up Guide*. Sharp Corporation, 2003

[10] *Qt Extended Whitepaper*. Nokia Corporation, 2008

[11] *ARM920T Technical Reference Manual*. ARM Limited, 2001

[12] *Data sheet*. Philips Semiconductors, 2001

[13] http://qt.nokia.com/

[14] http://es.wikipedia.org/

[15] http://en.wikipedia.org/

[16] http://equallybad.blogspot.com/

[17] http://www.friendlyarm.net/

[18] http://usemoslinux.blogspot.com/

[17] http://www.fsf.org/

[18] http://www.gnu.org/

[19] http://www.opensource.org/

[20] http://creativecommons.org/

ANEXO: LICENCIA GNU GPL

LICENCIA PÚBLICA GENERAL DE GNU[5]

Versión 3, 29 de junio de 2007

Preámbulo

La Licencia Pública General de GNU (GNU GPL, por sus siglas en inglés) es una licencia libre y gratuita con derecho de copia para software y otros tipos de obras.

Las licencias para la mayoría del software y otras obras de índole práctica están diseñadas para privarle de la libertad para distribuir y modificar las obras. Por el contrario, la Licencia Pública General de GNU garantiza la libre distribución y modificación de todas las versiones de un programa, a fin de asegurarle dicha libertad a todos los usuarios. En la Fundación para el Software Libre utilizamos la Licencia Pública General de GNU para la mayoría de nuestro software; también se aplica a cualquier otra obra publicada de esta manera por sus autores. Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre, nos referimos a la libertad, no al precio. Nuestras Licencias Públicas Generales están diseñadas para garantizarle a usted la libertad de distribuir copias de software libre (y cobrar por ellas, si así lo desea), obtener el código fuente, o tener la posibilidad de obtenerlo, modificar el software o utilizar partes del mismo en nuevos programas libres, y saber que puede hacer estas cosas.

Para proteger sus derechos, necesitamos evitar que otros le nieguen estos derechos o le pidan que renuncie a los mismos. Por lo tanto, en el caso de que usted distribuya o modifique este software, tendrá ciertas responsabilidades a fin de garantizar la libertad de los demás.

Por ejemplo, si usted distribuye copias de un programa de esta naturaleza, ya sea en forma gratuita o a cambio de dinero, debe extender a los destinatarios del software las mismas libertades que le fueron otorgadas a usted. Debe asegurarse de que ellos también reciban o tengan la posibilidad de obtener el código fuente. Y debe mostrarles los presentes términos a fin de que conozcan sus derechos.

Los desarrolladores que utilizan la GNU GPL siguen dos pasos para proteger los derechos que usted recibe: (1) declarar los derechos de autor del software, y (2) ofrecerle esta Licencia para que usted pueda copiar, distribuir y/o modificar el software legalmente.

A fin de proteger a los desarrolladores y autores, la GPL explica claramente que nos se ofrecen garantías por este software libre. Por el bien de los usuarios y de los autores, la GPL exige que las versiones modificadas se identifiquen como tales, de modo que los

5 La presente es una traducción no oficial de la Licencia Pública General de GNU al español. No ha sido publicada por la Fundación para el Software Libre ni establece legalmente los términos de distribución para el software que se rija por la Licencia Pública General de GNU (GNU GPL, por sus siglas en inglés); la única que lo hace es la versión original en inglés de la GNU GPL. No obstante, el objetivo de esta traducción es ayudar a los hispanoparlantes a comprender mejor la GNU GPL.

problemas que puedan contener estas versiones no se atribuyan erróneamente a los autores de versiones anteriores.

Existen algunos dispositivos diseñados para negarles a los usuarios el acceso para instalar o ejecutar versiones modificadas del software que contienen, aunque el fabricante pueda hacerlo. Esto es esencialmente incompatible con el objetivo de proteger la libertad de los usuarios para modificar el software. El patrón sistemático de tal abuso se da en el área de productos para el uso por parte de individuos, precisamente un área en la cual se vuelve más inaceptable. Por lo tanto, hemos diseñado esta versión de la GPL a fin de prohibir la práctica para dichos productos. En el caso de que dichos problemas surgieran en otras esferas, hemos tomado los recaudos necesarios para extender esta disposición a dichas esferas en futuras versiones de la GPL, según se requiera para proteger la libertad de los usuarios.

Por último, todos los programas se ven amenazados constantemente por patentes de software. Los Estados no deberían permitirles a las patentes restringir el desarrollo y el uso de software en computadoras para fines generales, pero, en el caso de que esto suceda, deseamos evitar el riesgo especial de que las patentes que se apliquen a un programa libre efectivamente otorguen tal exclusividad. Para lograrlo, la GPL garantiza la imposibilidad del uso de las patentes para apropiarse de un programa y restringir dicha libertad.

A continuación, se incluyen los términos y condiciones particulares para la reproducción, distribución y modificación del software.

TÉRMINOS Y CONDICIONES

0. Definiciones.

Por “esta Licencia” se entiende la versión 3 de la Licencia Pública General de GNU.

El término “copyright” también se extiende a las leyes que protegen los derechos de autor para otros tipos de obras, tales como diseños de circuitos integrados sobre substrato semiconductor.

Por “el Programa” se entiende cualquier obra incluida en esta Licencia sobre la que se puedan ejercer derechos de autor. Para referirnos a cada licenciatario, utilizamos el término “usted”. Los “licenciatarios” y “destinatarios” pueden ser individuos u organizaciones.

Por “modificar” una obra se entiende el proceso de copiar o adaptar una obra en forma parcial o total de un modo que requiera autorización de copyright y que no sea la reproducción de una copia exacta. La obra resultante es una “versión modificada” de la obra anterior o una obra “basada en” la obra anterior.

Por “obra amparada” se entiende el Programa sin modificaciones o una obra basada en el Programa.

Por “propagar” una obra se entiende cualquier acción sobre la misma que, en el caso de no tener autorización, pudiera hacerlo responsable, ya sea en forma directa o indirecta, de infringir las leyes de derechos de autor aplicables, salvo que dicha acción se realice en una computadora o se modifique una copia privada. La propagación incluye la reproducción, distribución (con o sin modificaciones), divulgación y, en algunos países, otras actividades también.

Por “transmitir” una obra se entiende cualquier tipo de propagación que le permita a un tercero hacer o recibir copias. La mera interacción con un usuario a través de una red informática, cuando no se transfiere una copia, no se considera una transmisión.

Una interfaz de usuario interactiva muestra “avisos legales apropiados” en la medida en que, de un modo práctico y bien visible, (1) muestre un aviso de copyright apropiado y (2) le informe al usuario que no se ofrecen garantías por la obra (salvo que efectivamente se ofrezcan garantías) y que los licenciatarios pueden transmitir la obra conforme a las disposiciones de esta Licencia, además de mostrar la forma en que se puede consultar una copia de esta Licencia. Si la interfaz presenta una lista de comandos del usuario u opciones, tales como un menú, dicha lista debe incluir un ítem visible que cumpla con este criterio.

1. Código fuente.

El “código fuente” de una obra es el formato preferido de la obra para realizar modificaciones en la misma. Por “código objeto” se entiende cualquier formato de una obra que no sea código fuente.

Una “interfaz estándar” es una interfaz que puede ser una norma oficial, según lo defina un organismo de normas reconocido, o bien, en el caso de interfaces específicas para un lenguaje de programación particular, una interfaz de uso generalizado entre los desarrolladores que trabajan con dicho lenguaje.

Las “bibliotecas de sistemas” de una obra ejecutable comprenden cualquier cosa, salvo la obra en su totalidad, que (a) se incluya en la forma normal de empaquetamiento de un Componente Importante, pero que no forme parte del Componente Importante, y (b) sirva únicamente para permitir el uso de la obra con dicho Componente Importante o para implementar una Interfaz Estándar para la cual haya a disposición del público una implementación en forma de código fuente. Un “Componente Importante”, en este contexto, es un componente fundamental (núcleo, sistema de ventanas, etc.) del sistema operativo específico (si hubiera) en el que funcione la obra ejecutable, o un compilador utilizado para producir la obra, o un intérprete de código objeto utilizado para ejecutarlo.

La “Fuente Correspondiente” para una obra en código objeto refiere a todo el código fuente necesario para generar, instalar y (para una obra ejecutable) ejecutar el código objeto y modificar la obra, incluidas las secuencias de comandos para controlar dichas actividades. Sin embargo, no incluye las Bibliotecas de Sistemas de la obra, así como tampoco herramientas de aplicación general o programas libres generalmente disponibles que se utilicen sin modificaciones para realizar dichas actividades pero que no formen parte de la obra. Por ejemplo, la Fuente Correspondiente incluye los archivos de definición de interfaz asociados a los archivos fuente para la obra, así como el código fuente para las bibliotecas compartidas y los subprogramas vinculados en forma dinámica requeridos específicamente conforme a su diseño, por ejemplo, mediante la comunicación de datos intrínseca o el control de flujo entre esos subprogramas y otras partes de la obra.

La Fuente Correspondiente no necesita incluir nada que los usuarios puedan regenerar automáticamente de otras partes de la Fuente Correspondiente.

La Fuente Correspondiente para una obra en código fuente es esa misma obra.

2. Permisos básicos.

Todos los derechos que se otorgan conforme a esta Licencia se otorgan por el término del

copyright que ampara al Programa y son irrevocables siempre y cuando se cumplan las condiciones establecidas. Esta Licencia lo autoriza en forma expresa e ilimitada a ejecutar el Programa sin modificaciones. El producto obtenido a partir de la ejecución de una obra amparada está cubierto por esta Licencia únicamente si el producto, dado su contenido, constituye una obra amparada. Esta Licencia reconoce sus derechos de uso razonable y otros equivalentes, conforme a las leyes de copyright.

Usted puede crear, ejecutar y propagar obras amparadas que no transmita, sin condiciones en la medida en que su licencia siga vigente de alguna otra manera. Usted puede transmitir obras amparadas a terceros con el único fin de que éstos realicen modificaciones exclusivamente para usted, o que le proporcionen los medios para ejecutar dichas obras, siempre y cuando usted cumpla con los términos de esta Licencia en lo que respecta a la transmisión de cualquier material que exceda su control del copyright. Aquéllos que de esta manera creen o ejecuten las obras aparadas para usted deben hacerlo exclusivamente en su nombre, bajo su dirección y control y sobre la base de términos que les prohíban hacer copias de su material protegido por derechos de autor fuera de la relación que mantienen con usted.

La transmisión bajo cualquier otra circunstancia se permite únicamente conforme a las condiciones que se describen a continuación. Se prohíbe sublicenciar; la sección 10 hace que sea innecesario.

3. Protección de los derechos legales de los usuarios frente a la Ley Anti-Evasión.

Ninguna obra amparada se considerará parte de una medida tecnológica efectiva conforme a cualquier ley aplicable que cumpla las obligaciones del artículo 11 del tratado de copyright WIPO adoptado el 20 de diciembre de 1996 o a leyes similares que prohíban o restrinjan la evasión de dichas medidas.

Cuando usted transmite una obra amparada, renuncia a cualquier facultad legal de prohibir la evasión de medidas tecnológicas en la medida en que dicha evasión se realice al hacer uso de los derechos que se otorgan conforme a esta Licencia con respecto a la obra amparada, y niega cualquier intención de restringir el uso o la modificación de la obra como una forma de hacer valer, en contra de los usuarios de la obra, sus derechos legales o los derechos legales de terceros para prohibir la evasión de medidas tecnológicas.

4. Transmisión de copias exactas.

Usted puede transmitir copias exactas del código fuente del Programa tal cual lo reciba, en cualquier medio, siempre y cuando publique de un modo llamativo y adecuado un aviso de copyright apropiado en cada copia; mantenga intactos todos los avisos que establecen que esta Licencia y cualquier término no-permisivo que se agregue conforme a la sección 7 se aplican al código; mantenga intactos todos los avisos mediante los cuales se niega cualquier tipo de garantía; y les proporcione a todos los destinatarios una copia de esta Licencia junto con el Programa.

Usted puede cobrar el precio que usted desee o no cobrar nada por cada copia que transmita, y puede ofrecer soporte o protección de garantía a cambio de una tarifa.

5. Transmisión de versiones modificadas del código fuente.

Usted puede transmitir una obra basada en el Programa, o las modificaciones para

producirlo a partir del Programa, en forma de código fuente conforme a los términos de la sección 4, siempre y cuando también cumpla con todas las condiciones que se incluyen a continuación:

- a) La obra debe conservar avisos llamativos que establezcan que usted la ha modificado e incluyan la fecha correspondiente.
- b) La obra debe conservar avisos llamativos que establezcan que la misma se realiza conforme a esta Licencia y a todas las condiciones que se agreguen bajo la sección 7. Este requerimiento modifica el requerimiento de la sección 4 que establece que se deben “mantener intactos todos los avisos”.
- c) Usted debe otorgar una licencia por la obra completa, en forma íntegra, conforme a esta Licencia, a cualquier tercero que adquiera una copia. Por lo tanto, esta Licencia, junto con cualquier término adicional aplicable de la sección 7, se aplica a la obra en su totalidad y a todas sus partes, independientemente del modo en que se las empaquete. Esta Licencia no lo autoriza a otorgar licencias para la obra de ningún otro modo, pero no invalida dicha autorización si usted la ha recibido por separado.
- d) Si la obra tuviera interfaces de usuario interactivas, cada una de ellas deberá mostrar Avisos Legales Apropiados. No obstante, si el Programa tuviera interfaces interactivas que no mostraran Avisos Legales Apropiados, usted no necesita incluirlos.

Se denomina “conjunto” a la compilación de una obra amparada con otras obras diferentes e independientes que por su naturaleza no sean extensiones de la obra amparada ni se combinen con ella para formar un programa más grande en un volumen de un medio de distribución o almacenamiento, si la compilación y el copyright consiguiente no se utilizan para restringir el acceso o los derechos legales de los usuarios de la compilación más allá de lo que permitan las obras individuales. La inclusión de una obra amparada en un conjunto no implica que esta Licencia se aplique a las otras partes del conjunto.

6. Transmisión de códigos que no son códigos fuente.

Usted puede transmitir una obra amparada en código objeto conforme a los términos de las secciones 4 y 5, siempre y cuando también transmita la Fuente Correspondiente legible por máquina conforme a los términos de esta Licencia, de alguna de las siguientes maneras:

- a) Transmisión del código objeto dentro de un producto físico (incluidos medios físicos de distribución) o incorporado a éste, acompañado de la Fuente Correspondiente en un medio físico duradero habitual para el intercambio de software.
- b) Transmisión del código objeto dentro de un producto físico (incluidos medios físicos de distribución) o incorporado a éste, acompañado de una oferta escrita, que sea válida por un plazo mínimo de tres años y por el tiempo que usted ofrezca repuestos o soporte técnico para ese modelo del producto, para proporcionarle a cualquier persona que posea el código objeto (1) una copia de la Fuente Correspondiente para todo el software del producto que esté amparado por esta Licencia, en un medio físico duradero habitual para el intercambio de software, a cambio de un precio que no exceda el costo razonable de la acción física de transmitir esta fuente, o (2) acceso para la copia de la Fuente Correspondiente desde un servidor de red sin costo alguno.

- c) Transmisión de copias individuales del código objeto junto con una copia de la oferta escrita para proporcionar la Fuente Correspondiente. Esta opción se permite únicamente en ocasiones y para fines no comerciales, y sólo en la medida en que usted haya recibido el código objeto con una oferta de esta naturaleza, conforme a la subsección 6b.
- d) Transmisión del código objeto ofreciendo acceso desde un lugar determinado (en forma gratuita u onerosa) y ofreciendo un acceso equivalente a la Fuente Correspondiente del mismo modo y desde el mismo lugar sin costo adicional. No es necesario que les exija a los destinatarios que copien la Fuente Correspondiente junto con el código objeto. Si el lugar ofrecido para copiar el código objeto fuera un servidor de red, la Fuente Correspondiente podrá estar en un servidor diferente (operado por usted o un tercero) que ofrezca posibilidades de reproducción equivalentes, siempre y cuando se incluyan, junto al código objeto, instrucciones claras para localizar la Fuente Correspondiente. Independientemente de qué servidor albergue la Fuente Correspondiente, usted mantiene la obligación de asegurarse de que el mismo esté disponible por el tiempo que sea necesario para satisfacer estos requerimientos.
- e) Transmisión del código objeto mediante transferencia entre usuarios (peer to peer), siempre y cuando les informe a los usuarios la ubicación del código objeto y la Fuente Correspondiente de la obra para el público en general sin costo alguno conforme a la subsección 6d.

No se necesita incluir una parte separable del código objeto, cuyo código fuente se excluya de la Fuente Correspondiente como una biblioteca de sistemas, para transmitir el código objeto de la obra.

Por “producto de usuario” se entiende (1) un “producto de consumo”, que es cualquier bien personal tangible que se utilice habitualmente para fines personales, familiares o domésticos, o (2) cualquier cosa que se diseñe o comercialice para su incorporación en una vivienda. Al determinar si un producto es un producto de consumo, los casos dudosos deberán resolverse a favor del amparo. Para un producto específico que recibe un usuario particular, un “uso habitual” es el uso común o típico que se le suele dar a ese tipo de producto, independientemente de la condición del usuario particular o de la forma en que el usuario particular utilice el producto o de las expectativas propias o de terceros con respecto al uso del producto. Un producto se considera un producto de consumo independientemente de que que se le pueda dar usos sustanciales de índole comercial, industrial o ajena al consumo, salvo que dichos usos representen el único modo significativo de utilizar el producto.

Por “información de instalación” de un producto de usuario se entiende cualquier método, procedimiento, clave de autorización u otro tipo de información requerida para instalar y ejecutar versiones modificadas de una obra amparada en dicho producto de usuario a partir de una versión modificada de su Fuente Correspondiente. La información debe ser suficiente para garantizar que el funcionamiento continuo del código objeto modificado no se vea afectado o imposibilitado por el solo hecho de haberse realizado la modificación.

En el caso de que usted transmita el código objeto de una obra conforme a esta sección en un producto de usuario, junto con un producto de usuario o específicamente para su uso en un producto de usuario, y la transmisión se produzca como parte de una transacción mediante la cual los derechos de posesión y uso del producto de usuario se transfieran al destinatario por un plazo limitado o ilimitado (independientemente de las particularidades de la transacción), la Fuente Correspondiente transmitida conforme a

esta sección deberá ir acompañada de la información de instalación. Sin embargo, este requerimiento no se aplicará en el caso de que ni usted ni un tercero conserven la capacidad para instalar el código objeto modificado en el producto de usuario (por ejemplo, que la obra se haya instalado en memoria ROM).

El requerimiento de proporcionar información de instalación no implica la necesidad de seguir proveyendo soporte técnico, garantías o actualizaciones para una obra que haya sido modificada o instalada por el destinatario o para el producto de usuario en el cual se la haya modificado o instalado. Podrá negarse el acceso a una red cuando la modificación en sí misma pueda afectar de un modo adverso y sustancial el funcionamiento de la red o infrinja las normas y los protocolos de comunicación a través de la red.

La Fuente correspondiente que se transmita y la información de instalación que se proporcione conforme a esta sección deberán presentarse en un formato sobre el cual exista documentación pública (y con una implementación disponible para el público en código fuente) y no deberán requerir ninguna clave o contraseña especial para su desempaquetamiento, lectura o reproducción.

7. Términos adicionales.

Los “permisos adicionales” son términos que complementan los términos de esta Licencia al permitir excepciones a una o más condiciones. Los permisos adicionales que se aplican al Programa en su totalidad deberán tratarse como si formaran parte de esta Licencia, en la medida en que sean válidos conforme a las leyes aplicables. En el caso de que los permisos adicionales se apliquen únicamente a una parte del Programa, esta parte podrá utilizarse por separado conforme a dichos permisos, pero el Programa en su totalidad seguirá rigiéndose de acuerdo a esta Licencia independientemente de los permisos adicionales.

Cuando usted transmita una copia de una obra amparada, podrá optar por eliminar cualquier permiso adicional de dicha copia o de cualquier parte de la misma (en ciertos casos, cuando usted modifique la obra, podrán establecerse permisos adicionales para requerir la eliminación de los mismos). Tiene autorización para incluir permisos adicionales en un material que usted haya agregado a una obra amparada y sobre el cual usted posea o pueda otorgar permisos de copyright adecuados.

Independientemente de cualquier otra disposición de esta Licencia, con respecto al material que usted agregue a una obra amparada, usted podrá (en la medida en que lo autoricen los titulares de los derechos de copyright de dicho material) complementar los términos de esta Licencia con los siguientes términos:

- a) Ausencia de garantías o limitación de la responsabilidad más allá de los términos de las secciones 15 y 16 de esta Licencia; o
- b) Obligación de conservación de atribuciones de autoría o avisos legales razonables específicos en dicho material o en los avisos legales apropiados que se muestren en las obras que lo contengan; o
- c) Prohibición de tergiversación del origen del material, o requerimiento de que en las versiones modificadas de dicho material se indique de un modo razonable que son diferentes de la versión original; o
- d) Limitación del uso de los nombres de los licenciantes o autores del material para fines publicitarios; o
- e) Negativa con respecto al otorgamiento de derechos conforme a las leyes de marcas para el uso de ciertos nombres comerciales, marcas de productos o marcas

de servicios; o

- f) Requerimiento de indemnización de los licenciantes o autores de dicho material por parte de cualquier persona que transmita el material (o versiones modificadas del mismo) bajo presunciones contractuales de responsabilidad del destinatario por cualquier responsabilidad que dichas presunciones contractuales impongan directamente sobre los licenciantes y autores del material.

Cualquier otro término adicional no permisivo se considerará una “restricción adicional” en el contexto de la sección 10. En el caso de que el Programa, tal cual usted lo recibió, o cualquier parte del mismo contengan un aviso que indique que el mismo se rige según esta Licencia junto con un término que constituya una restricción adicional, podrá eliminar dicho término. En el caso de que un documento de licencia contenga una restricción adicional pero permita la extensión de la licencia o la transmisión del programa conforme a esta Licencia, usted podrá agregar a la obra amparada cualquier material conforme a los términos de dicho documento de licencia, siempre y cuando la restricción adicional no se mantenga tras la extensión de la licencia o la transmisión del programa.

En el caso de que usted agregue términos a una obra amparada conforme a esta sección, deberá incluir en los archivos fuente correspondientes una declaración de los términos adicionales que se aplican a dichos archivos o un aviso que indique la ubicación de los términos aplicables.

Se podrán establecer términos adicionales, sean éstos permisivos o no permisivos, en una licencia escrita independiente, o a modo de excepciones; sea como fuere, se aplicarán los requerimientos mencionados anteriormente.

8. Cancelación.

Usted no está autorizado a propagar o modificar una obra amparada de ningún otro modo que no se estipule en esta Licencia. Cualquier intento no autorizado por propagarla o modificarla se considerará nulo y conllevará la cancelación automática de los derechos que le haya otorgado esta Licencia (incluida cualquier licencia de patente otorgada conforme al párrafo tercero de la sección 11).

No obstante, en el caso de que deje de violar las cláusulas de esta Licencia, un titular de derechos de copyright particular podrá restituirle la licencia (a) en forma provisoria, hasta tanto dicho titular dé por finalizada su licencia en forma expresa y definitiva, y (b) en forma permanente, si dicho titular no lo notificara de la infracción por algún medio razonable antes de los 60 días posteriores a la cancelación.

Asimismo, la licencia que le otorgue un titular de derechos de copyright particular se le restituirá en forma permanente si dicho titular lo notificara de la infracción por algún medio razonable, ésta fuera la primera vez que usted hubiese recibido una notificación de violación de esta Licencia (por cualquier obra) emitida por dicho titular, y usted subsanara la infracción en un plazo de 30 días a partir de la recepción de la notificación.

La extinción de sus derechos conforme a esta sección no cancela las licencias de aquellos terceros a los que usted les haya otorgado copias o derechos conforme a esta Licencia. En el caso de que sus derechos se cancelen y no se le restituyan en forma permanente, usted no estará capacitado para recibir nuevas licencias para el mismo material conforme a la sección 10.

9. Aceptación innecesaria para la posesión de copias.

Usted no está obligado a aceptar esta Licencia para poder recibir o ejecutar una copia del Programa. De modo similar, la propagación secundaria de una obra amparada que se produzca únicamente como consecuencia de una transferencia entre usuarios (peer to peer) a fin de recibir una copia tampoco requiere aceptación. No obstante, esta Licencia es lo único que lo autoriza a propagar o modificar cualquier obra amparada. En el caso de que usted no acepte esta Licencia, los actos anteriores representarán una violación de las leyes de copyright. Por lo tanto, al modificar o propagar una obra amparada, usted expresa su aceptación de esta Licencia para hacerlo.

10. Traspaso automático de licencia a destinatarios subsiguientes.

Cada vez que usted transmite una obra amparada, el destinatario recibe automáticamente de los licenciantes originales una licencia para ejecutar, modificar y propagar la obra conforme a esta Licencia. Usted no es responsable de asegurar el cumplimiento de esta Licencia por parte de terceros.

Una “transacción entre entidades” es una transacción mediante la cual se transfiere el control de una organización o de todo el patrimonio de una organización, se subdivide una organización o se fusionan dos o más organizaciones. En el caso de que la propagación de una obra amparada se deba a una transacción entre entidades, cada parte de la transacción que reciba una copia de la obra también recibirá todas las licencias para la obra que el predecesor de la parte tuviera o pudiera otorgar conforme al párrafo anterior, más el derecho de recibir de su predecesor la Fuente Correspondiente de la obra, si el predecesor la tuviera en su poder o pudiera obtenerla con un esfuerzo razonable.

Usted no puede imponer restricciones adicionales para el ejercicio de los derechos que se otorgan o consolidan conforme a esta Licencia. Por ejemplo, usted no puede imponer tarifas, regalías u otros cargos a cambio del ejercicio de los derechos que se otorgan conforme a esta Licencia, así como tampoco puede iniciar acciones legales (incluidas demandas y contrademandas en un pleito) sobre la base de una infracción de patentes por crear, usar, comercializar, ofrecer para la venta o importar el Programa o cualquier parte del mismo.

11. Patentes.

Un “colaborador” es un titular de derechos de copyright que autoriza el uso conforme a esta Licencia del Programa o de una obra sobre la cual se base el Programa. La obra cuya licencia se otorgue de esta manera se denomina “versión en colaboración” del colaborador.

Los “derechos de patente fundamentales” de un colaborador son todos los derechos de patente bajo la titularidad o el control del colaborador, ya sea que se los hubiese adquirido previo al otorgamiento de esta Licencia o a partir del mismo, que puedan infringirse de algún modo permitido por esta Licencia para crear, usar o vender su versión en colaboración, pero no incluyen derechos que se podrían infringir únicamente como consecuencia de modificaciones posteriores a la versión en colaboración. A los efectos de esta definición, el “control” incluye el derecho de otorgar sublicencias de patente de un modo acorde a los requerimientos de esta Licencia.

Cada colaborador le otorga a usted una licencia de patente internacional no-exclusiva libre de regalías conforme a los derechos de patente fundamentales del colaborador para crear, usar, comercializar, ofrecer para la venta, importar y ejecutar, modificar y propagar

de algún otro modo el contenido de su versión en colaboración.

En los tres párrafos que se incluyen a continuación, una “licencia de patente” es cualquier contrato o acuerdo expreso, independientemente de su denominación, mediante el cual se convenga no ejercer derechos de patente (como, por ejemplo, una autorización expresa para hacer uso de una patente o una cláusula que establezca que no se iniciarán acciones legales por infringir los derechos de patente). Por “otorgar” una licencia de patente de esta naturaleza a otra parte se entiende el acto de celebrar un contrato o acuerdo mediante el cual se conviene no ejercer derechos de patente en contra de dicha parte.

En el caso de que usted transmita una obra amparada, a sabiendas de que está sujeta a una licencia de patente, y la Fuente Correspondiente de la obra no estuviera disponible para su reproducción, en forma gratuita y conforme a los términos de esta Licencia, a través de un servidor de red de acceso público u otro medio igualmente accesible, usted deberá (1) poner la Fuente Correspondiente a disposición del destinatario subsiguiente, (2) renunciar al beneficio de la licencia de patente para esta obra en particular, o bien (3) tomar las medidas necesarias para extender la licencia de patente a los destinatarios subsiguientes de un modo acorde a los requerimientos de esta Licencia. La frase “a sabiendas de que está sujeta a una licencia de patente” significa que usted efectivamente sabe que, de no ser por la licencia de patente, su transmisión de la obra amparada en un país o el uso que pudiera darle el destinatario a la obra amparada en un país infringirían una o más patentes identificables en dicho país que usted considera válidas por diversas razones.

En el caso de que, en relación con una transacción o contrato individual, usted transmitiera una obra amparada o la propagara consiguiendo su transmisión y otorgara a algunas de las partes que reciban la obra amparada una licencia de patente que las autorizara a usar, propagar, modificar o trasmitir una copia específica de la obra amparada, la licencia de patente que usted otorgue se extenderá automáticamente a todos los destinatarios de la obra amparada y a las obras que se basen en ella.

Una licencia de patente se considera “discriminatoria” cuando no incluye dentro de su ámbito de cobertura uno o más de los derechos que se otorgan específicamente conforme a esta Licencia, prohíbe el uso de dichos derechos o se otorga como condición de que no se usen dichos derechos. Usted no debe transmitir una obra amparada si fuese una de las partes de un contrato con un tercero que se dedicara a la distribución de software, conforme al cual usted debiera pagarle al tercero por la actividad que usted realice con respecto a la transmisión de la obra y el tercero le otorgara a cualquiera de las partes que reciban de usted la obra amparada una licencia de patente discriminatoria (a) en relación con las copias de la obra amparada transmitidas por usted (o las copias que se hagan de esas copias), o (b) principalmente para compilaciones o productos específicos que contengan la obra amparada y en relación con éstos, a menos que usted hubiese celebrado dicho contrato o que la patente se hubiese otorgado antes del 28 de marzo de 2007.

Ninguna disposición de esta Licencia deberá interpretarse como excluyente o limitativa de ninguna licencia implícita u otras defensas legales contra infracciones a las que, de otro modo, usted pudiese tener derecho conforme a la ley de propiedad intelectual vigente.

12. Protección de la libertad de terceros.

En el caso de que le fueran impuestas condiciones (ya sea por una orden judicial, un

contrato o de algún otro modo) que contradijeran las condiciones de esta Licencia, usted no quedará eximido de cumplir las condiciones de esta Licencia. En el caso de que no pueda transmitir una obra amparada de un modo que le permita cumplir simultáneamente con las obligaciones establecidas por esta Licencia y cualquier otra obligación pertinente, no podrá transmitirla de ningún modo. Por ejemplo, en el caso de que usted acepte términos que lo obliguen a cobrar regalías por retransmisión de aquéllos a los que usted transmita el Programa, la única forma de satisfacer tanto dichos requerimientos como esta Licencia será abstenerse de transmitir el Programa.

13. Uso conjunto con la Licencia Pública General Affero de GNU.

Independientemente de cualquier otra disposición de esta Licencia, usted tiene permiso para vincular o combinar cualquier obra amparada con una obra cuya licencia se otorgue conforme a la versión 3 de la Licencia Pública General Affero de GNU en una única obra combinada y transmitir la obra resultante. Los términos de esta Licencia seguirán aplicándose a la parte que corresponda a la obra amparada, pero los requerimientos especiales de la sección 13 de la Licencia Pública General Affero de GNU sobre la interacción a través de una red se aplicarán a la combinación como tal.

14. Revisiones de esta Licencia.

La Fundación para el Software Libre podrá publicar revisiones y/o versiones nuevas de la Licencia Pública General de GNU de vez en cuando. Tales versiones serán de naturaleza similar a la versión actual, pero podrán diferir en cuanto a los detalles para afrontar nuevos problemas o inquietudes.

Cada versión recibirá un número de versión que la distinga. En el caso de que el Programa especifique que se rige por una versión determinada de la Licencia Pública General de GNU “o cualquier versión posterior”, usted podrá optar por adoptar los términos y condiciones de dicha versión específica o de cualquier versión posterior que publique la Fundación para el Software Libre. En el caso de que el Programa no especifique un número de versión de la Licencia Pública General de GNU, usted podrá regirse por cualquier versión que haya publicado la Fundación para el Software Libre.

Si el Programa especificara que un apoderado puede decidir qué versiones de la Licencia Pública General de GNU pueden aplicarse en el futuro, la declaración pública del apoderado sobre la aceptación de una versión determinada lo autorizará a usted, en forma permanente, a optar por dicha versión para el Programa.

Puede que las versiones posteriores de la licencia le otorguen permisos adicionales o diferentes. No obstante, no se les impondrán obligaciones adicionales a ningún autor o titular de derechos de copyright como resultado de la adopción de la versión posterior que usted elija.

15. Ausencia de garantías.

EL PROGRAMA SE OFRECE SIN NINGÚN TIPO DE GARANTÍAS, EN LA MEDIDA EN QUE LO PERMITAN LAS LEYES APLICABLES. SALVO DISPOSICIÓN CONTRARIA POR ESCRITO, LOS TITULARES DE DERECHOS DE COPYRIGHT Y/U OTRAS PARTES PROVEEN EL PROGRAMA "TAL CUAL" SIN GARANTÍAS DE NINGÚN TIPO, YA SEAN EXPRESAS O IMPLÍCITAS, INCLUIDAS, AUNQUE NO EN FORMA TAXATIVA, LAS GARANTÍAS IMPLÍCITAS DE COMERCIABILIDAD Y APTITUD PARA UN PROPÓSITO DETERMINADO. USTED ASUME TODOS LOS RIESGOS CON

RESPECTO A LA CALIDAD Y EL DESEMPEÑO DEL PROGRAMA. EN EL CASO DE QUE EL PROGRAMA TUVIERA DEFECTOS, USTED ASUME EL COSTO DE TODAS LAS ACTIVIDADES DE MANTENIMIENTO, REPARACIÓN O CORRECCIÓN.

16. Limitación de la responsabilidad.

EN NINGÚN CASO, SALVO QUE ASÍ LO DISPONGAN LAS LEYES APLICABLES O UN CONTRATO POR ESCRITO, UN TITULAR DE DERECHOS DE COPYRIGHT O UN TERCERO QUE MODIFIQUE Y/O TRANSMITA EL PROGRAMA SEGÚN SE AUTORIZA ANTERIORMENTE SERÁ RESPONSABLE ANTE USTED DE CUALQUIER DAÑO, INCLUIDOS DAÑOS GENERALES, ESPECIALES, FORTUITOS O DERIVADOS, QUE PUEDA SURGIR DEL USO O LA INCAPACIDAD DE USO DEL PROGRAMA (INCLUIDOS, AUNQUE NO TAXATIVAMENTE, LA PÉRDIDA DE INFORMACIÓN, EL SUMINISTRO DE INFORMACIÓN IMPRECISA O LAS PÉRDIDAS QUE PUEDAN SUFRIR USTED O UN TERCERO O LA INCAPACIDAD DEL PROGRAMA PARA INTERACTUAR CON OTROS PROGRAMAS), AUN CUANDO DICHO TITULAR O TERCERO HUBIESE SIDO ADVERTIDO DE LA POSIBILIDAD DE TALES DAÑOS.

17. Interpretación de las secciones 15 y 16.

En el caso de que las cláusulas de ausencia de garantías y limitación de la responsabilidad anteriores carecieran de validez legal a nivel local conforme a sus términos, los juzgados deberán aplicar las leyes locales que más se asimilen a una exención absoluta de cualquier responsabilidad civil en relación con el Programa, salvo que una copia del Programa estuviera acompañada de una garantía o presunción de responsabilidad a cambio de una tarifa.

FIN DE TÉRMINOS Y CONDICIONES